

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-021059

(43)Date of publication of application : 23.01.1998

(51)Int.Cl. G06F 9/06

G06F 13/00

G06F 13/00

(21)Application number : 08-172271 (71)Applicant : MITSUBISHI ELECTRIC CORP

(22)Date of filing : 02.07.1996 (72)Inventor : MOMOMOTO MASAHIRO KANAMORI TAKURO

(54) SOFTWARE VERSION MANAGEMENT SYSTEM FOR NETWORK SYSTEM

(57)Abstract:

PROBLEM TO BE SOLVED: To surely match the versions of software of an execution client with each other, to prevent inconsistency due to version mismatching, and to actualize stable system operation by replacing a client program of mismatching version on the client side.

SOLUTION: On the side of a client machine 2, it is judged whether the version of a client program 13 matches with that of a server program 14 according to a version check result transferred from a server machine 1. When it is judged that they do not match with each other, the client program 13 is replaced so that its version will match the version of the server program 14 and a replacing client program 13 is executed. Consequently, when the client program 13 is executed, its version can

surely be matched with the version of the server program 14.

LEGAL STATUS [Date of request for examination] 20.12.1996

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number] 3119166

[Date of registration] 13.10.2000

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right] 13.10.2003

* NOTICES *

JPO and NCIPI are not responsible for any damages caused by the use of this translation.

1. This document has been translated by computer. So the translation may not reflect the original precisely.

2.**** shows the word which can not be translated.

3. In the drawings, any words are not translated.

CLAIMS

[Claim(s)]

[Claim 1] In the software version control method of the network system by which the server was connected with the client through the network A version information setting means to set up the version information of a client program required for a version check, A version check request means to transmit the set-up version information to a server and to perform the version check request of a client program is formed in a client. The version information of the client program transmitted from a client is received. A version check request receptionist means to receive a version check request, A version information storing means to store the version information of a server program, The version information of the client program which received is compared with the version information of the server program read from the version

information storing means. A version check means to perform the version check of a client program and a server program, A version check result transfer means to transmit a version check result to a client is formed in a server. A version inequality decision means to judge whether the version of a client program is in agreement with the version of a server program based on the version check result transmitted from a server, A program exchange means to replace the client program whose version does not correspond so that it may be in agreement with the version of a server program when it is judged that the version is not in agreement, The software version control method of the network system characterized by forming a program execution means to perform the replaced client program in a client.

[Claim 2] It is the software version control method of the network system according to claim 1 characterized by to form a timing assignment means specify the timing of a version check before performing a version check request, and a timing decision means judge whether the timing of the specified version check was reached, in a client, to transmit the version information of the client program set up when it was judged that said version check request means reached the timing of the specified version check to a server, and to perform a version check request.

[Claim 3] It is the software version control method of a network system given in any of claims 1 and 2 characterized by said version check means performing a version check based on the set-up version check information form a check information setting means to set up the version check information that it uses for a version check before performing a version check in a server, and they are.

[Claim 4] It is the software version control method of a network system given in any of claims 1 and 2 characterized by said version check means performing a version check based on the set-up check logic form a logic setting means to set up the check logic at the time of a version check in a server, and they are, before performing a version check.

[Claim 5] It is the software version control method of a network system given in any of claims 1 and 2 characterized by said version check means performing a version check based on the dependency between the set-up related software form a dependency setting means to set up the dependency between related software before performing a version check in a server, and they are.

[Claim 6] It is the software version control method of a network system given in any of claims 1–5 characterized by replacing a client program when it is judged that said program exchange means reached the exchange timing of the set-up client program form a timing setting means to set up the timing which replaces a client program before replacing a client program, and a timing decision means to judge whether the exchange timing of the set-up client program was reached in a client, and they are.

[Claim 7] It is the software version control method of a network system given in any of claims 1–5 characterized by said program exchange means replacing a client program

based on the exchange conditions of the set-up client program form an exchange conditioning means to set up the exchange conditions of a client program in a client, and they are, before replacing a client program.

[Claim 8] It is the software version control method of a network system given in any of claims 1–5 characterized by the thing which chose said program exchange means, and for which it changes and the target client program is replaced form an exchange object-choice means to choose the client program for exchange in a client, and they are, before replacing a client program.

DETAILED DESCRIPTION

[Detailed Description of the Invention]

[0001]

[Field of the Invention] When the software version control method of a network system can be started, it can apply to the software version control method which operates on a distributed system in detail and software of a client is performed especially, this invention doubles the version of the software of the client to perform certainly, and relates to the software version control method of the network system which can realize system behavior stabilized in the mismatching by the version inequality as did not happen.

[0002]

[Description of the Prior Art] By the version control method of the conventional software which operates on a distributed system, the software installed in each machine by the database for version controls and its version information are managed intensively. To the version control method of this conventional software By the retry function at the time of the software distribution function and software distribution when setting up a schedule going wrong, after managing this software and its version information etc. The version information of the thing of the gestalt that the adjustment of the version of the software of each client will be guaranteed as an addition function of a software distribution function at the time of distribution, and the whole distributed system is managed. When the version of the software of each client is checked periodically and the inequality of a version is detected, in order to double a version, the thing of a gestalt which directs exchange of software to the client which a version does not suit is known.

[0003] First, a drawing is used and explained below about the version control method of the conventional software of the gestalt that coincidence of a system-wide version will be guaranteed by the retry function at the time of a former schedule function and software distribution going wrong. Drawing 8 is a flow chart which shows the system

behavior flow in the version control method of the conventional software. After a software distribution server sets up schedule information, such as a distribution place of the software to distribute, and exchange time amount, (step S801), according to the set-up schedule information, it performs message distribution processing of the software to the client of a distribution place, and distributes the file for software exchange to a client (step S802). At this time, information, such as exchange time amount of the software of the set-up schedule information, is also transmitted to a client.

[0004] A client will notify the receiving result of the file for software exchange to a software distribution server, if the file for software exchange distributed from the software distribution server is received (step S803) (step S804). A client carries out exchange processing of software according to information, such as exchange time amount of the software of the schedule information which it was transmitted from the software distribution server and received, when the file for software exchange distributed from the software distribution server is received and reception of the file for software exchange is performed normally (step S805).

[0005] A client notifies the result of software exchange processing to a software distribution server (step S806). A software distribution server judges whether the software distribution result of the file confirmation-of-receipt result for software exchange and a software exchange processing result notified from the client was received (step S807), and software distribution was normally processed based on the received software distribution result (step S808).

[0006] When it is judged that software distribution was not processed normally but the software distribution server has gone wrong (step S808), software is again distributed to a client, without changing the set-up schedule information (step S802). On the other hand, when it is judged that software distribution was processed normally (step S808), a software distribution server judges whether it changes to others and there is any object software (step S809), and in a certain case, return (step S801) is ended to processing of the following software, and when there is nothing, it ends processing.

[0007] By the version control method of this conventional software, when a client replaces software periodically or the time of the software distribution from a distribution server going wrong and a client fail in exchange of software according to the schedule information set up by the distribution server side, software is again distributed from a distribution server and exchange processing of software is performed.

[0008] Next, when a version is checked periodically [the latter mentioned above] and the inequality of a version is detected, a drawing is used and explained below about the version control method of the conventional software of the gestalt which directs exchange of software. Drawing 9 is a flow chart which shows the system behavior flow in the version control method of the conventional software. A client sets up

information required for a version check, and requests the version check of software to a version control server periodically based on the set-up information (step S901). At this time, a client transmits the version information of the software of a client, the version check request information on software, etc. to a version control server.

[0009] If the version check request of software is received from a client (step S902), based on the version information managed in the version control database, a version control server will check the version information of the software of a client, and will check the adjustment of a version (step S903). A version control server transmits information to a client as a result of a version check, and answers (step S904).

[0010] It judges whether exchange of whether if the version check result information transmitted from a version control server is received (step S905), the version of the software of a client is in agreement with the version of the software managed by the version control server based on the received version check result information, and software is required for a client (step S906).

[0011] A client will end processing as it is, if the version of the software of a client judges that the software of a client does not need to be changed in accordance with the version managed by the version control server (step S906). On the other hand, if the version of a client of the software of a client does not correspond with the version managed by the version control server but it judges that the software of a client needs to be changed (step S906), distribution of the file for software exchange will be required of a software distribution server (step S907).

[0012] A software distribution server sets up the schedule information which followed the software distribution demand from the client (step S908), and distributes the file for software exchange to the client of distribution demand origin based on the set-up schedule information (step S909). At this time, information, such as exchange time amount of the software of the set-up schedule information, is also transmitted to a client.

[0013] A client will carry out software exchange processing according to information, such as exchange time amount of the received schedule information, if the file for software exchange distributed from the software distribution server is received (step S910) (step S911).

[0014] By the version control method of this conventional software, if the version check of software is periodically required of a version control server from a client side and the version of a client is not in agreement with the version of a version control server, a client will get the software for exchange processing from a software distribution server, and will perform exchange processing of software.

[0015]

[Problem(s) to be Solved by the Invention] By the version control method of the above-mentioned former conventional software While a client can replace software periodically according to the schedule information set up by the distribution server

side Although software can be again distributed from a distribution server and exchange processing of software can be performed when the time of the software distribution from a distribution server going wrong and a client fail in exchange of software When a client program is performed before the software distribution completion to a client side, The version coincidence between server programs is not guaranteed, and exchange implementation of the software of a client cannot carry out to the timing of client program starting at the time of version inequality detection. For this reason, there was a problem of the mismatching by the version inequality having arisen and being hard to perform stable system behavior, at the time of software activation.

[0016] Moreover, by the version control method of the conventional software of this former, since it changed from a distribution server and may have stopped being able to get software of business when a network error etc. arises, there was a problem of the ability to stop being able to replace software.

[0017] By the version control method of the above-mentioned latter software The version check of software is periodically required of a version control server from a client side. Although a client can get the software for exchange processing from a software distribution server and can perform exchange processing of software when the version of a client is not in agreement with the version of a version control server Since software is not necessarily replaced when performing the version check of the software of a client periodically and performing software of a client, When performing software of a client, it is always hard to make a version in agreement. For this reason, there was a problem of the mismatching by the version inequality having arisen and being hard to perform stable system behavior, at the time of software activation.

[0018] Moreover, by the version control method of the software of this latter, simple coincidence retrieval is performed also about the method of a version check, and it was able to respond to neither exchange of the software with which compatibility was guaranteed between different versions, nor the complicated relation of the client application and the server from which two or more versions differ.

[0019] Moreover, by the version control method of the software of this latter, since it changed from a distribution server and may have stopped being able to get software of business when a network error etc. arises, there was a problem of the ability to stop being able to replace software.

[0020] Then, this invention doubles certainly the version of the software of the client performed when performing software of a client, and even if a network error arises, it aims at offering the software version control method of the network system with which the software of the client which was a version inequality can be replaced certainly, while it can realize system behavior stabilized in the mismatching by the version inequality as did not happen.

[0021]

[Means for Solving the Problem] In the software version control method of the network system by which, as for invention according to claim 1, the server was connected with the client through the network A version information setting means to set up the version information of a client program required for a version check, A version check request means to transmit the set-up version information to a server and to perform the version check request of a client program is formed in a client. The version information of the client program transmitted from a client is received. A version check request receptionist means to receive a version check request, A version information storing means to store the version information of a server program, The version information of the client program which received is compared with the version information of the server program read from the version information storing means. A version check means to perform the version check of a client program and a server program, A version check result transfer means to transmit a version check result to a client is formed in a server. A version inequality decision means to judge whether the version of a client program is in agreement with the version of a server program based on the version check result transmitted from a server, A program exchange means to replace the client program whose version does not correspond so that it may be in agreement with the version of a server program when it is judged that the version is not in agreement, It is characterized by forming a program execution means to perform the replaced client program in a client.

[0022] A timing assignment means to specify the timing of a version check before invention according to claim 2 performs a version check request in invention of the claim 1 above-mentioned publication, A timing decision means to judge whether the timing of the specified version check was reached is formed in a client. When said version check request means judges that the timing of the specified version check was reached, it is characterized by ***** which transmits the version information of the set-up client program to a server, and performs a version check request.

[0023] Invention according to claim 3 forms a check information setting means to set up the version check information that it uses for a version check before performing a version check in invention given in any of above-mentioned claims 1 and 2 they are in a server, and said version check means is characterized by performing a version check based on the set-up version check information.

[0024] Invention according to claim 4 forms a logic setting means to set up the check logic at the time of a version check before performing a version check in invention given in any of above-mentioned claims 1 and 2 they are in a server, and said version check means is characterized by performing a version check based on the set-up check logic.

[0025] Invention according to claim 5 forms a dependency setting means to set up the dependency between related software before performing a version check in invention given in any of above-mentioned claim 1 and two publications they are in a server, and

said version check means is characterized by performing a version check based on the dependency between the set-up related software.

[0026] When it is judged that invention according to claim 6 reached the exchange timing of the client program which formed a timing setting means to set up the timing which replaces a client program, and a timing decision means judged whether the exchange timing of the set-up client program was reached in the client, and said program exchange means set up before replacing a client program in invention given in any of above-mentioned claims 1-5 they are, it carries out changing a client program as the description.

[0027] In invention given in any of above-mentioned claims 1-5 they are, before invention according to claim 7 replaces a client program, it forms an exchange conditioning means to set up the exchange conditions of a client program in a client, and is characterized by said program exchange means replacing a client program based on the exchange conditions of the set-up client program.

[0028] In invention given in any of above-mentioned claims 1-5 they are, before invention according to claim 8 replaces a client program, it forms an exchange object-choice means to choose the candidate for exchange of a client program in a client, and is characterized by replacing a client program based on the candidate for exchange of the client program which said program exchange means chose.

[0029]

[Embodiment of the Invention] Hereafter, the gestalt of operation of this invention is explained with reference to a drawing.

Gestalt 1. drawing 1 of operation is the block diagram showing the system configuration of the software version control method of the network system of the gestalt 1 of operation concerning this invention. The machine configuration of a network system consists of the server machine 1 and client machine 2 of the number of arbitration, and each computer is connected to the network through the communication media 3, such as LAN. The server machine 1 has the magnetic disk drive 5 for storing the software package 4 which is a settlement of an application program, and the magnetic disk drive 7 for storing the version control database 6 which manages version information.

[0030] Next, a software configuration is explained. First, the software about the version control by the side of the server machine 1 consists of a version control database 6 which records the version information of each software on each network, a version control server program 8 which receives the version check request by the client machine 2, and an action description file 9 specified about the version check approach.

[0031] The software about the version control by the side of a client machine 2 consists of an action description file 11 which specifies actuation of the software before and behind the version check request to the version control server program 8,

and a version control client program 12 which publishes a version check request.

[0032] The applications which operate under a distributed-system environment are the application client program 13 and the application server program 14. Starting of a client program 13 is performed through the version control client program 12. Distribution of software and exchange of the software after a version check are performed when the distribution program 16 by the side of the server machine 1 sends a software package 4 to the distribution program 217 by the side of a client machine 2. [0033] Next, the configuration of an action description file is explained. Drawing 2 is drawing showing the configuration of the action description file 11 on the client machine 2 shown in drawing 1. The description 211 about the timing which performs a version check, and description 212 about the software exchange approach after it turns out that the version changes between a client program and a server program with version checks are performed in the action description file 11 on a client machine 2. As the software exchange approach, when software is replaced or which software being replaced, and information are defined.

[0034] Next, drawing 3 is drawing showing the configuration of the action description file 9 on the server machine 1 shown in drawing 1. Description 411 about the version check approach of software is performed in the action description file 9 on the server machine 1. a client side side/server side -- which action description files 9 and 11 -- "Product" -- 221, 321, and "Program" -- it has a tag name called 222 and 322.

[0035] "Program" means each application program and describes the program name which corresponds following on "Program." "Product" means the assembly of a program, and the assembly of this program is set up per business and usually describes a product name following on "Product." Each two or more "Product" "Program(s)" tag in a tag can be described. This is for offering the version check approach of a proper for every application program contained in "Product."

[0036] The configuration of the action description file 11 by the side of a client machine 2 consists of a part 214 which performs description about what does not use a default solution among the part 213 which specifies the default actuation about the conditions before a version check, and the processing after inequality detection about the corresponding product, and the program included in a certain product.

[0037] The configuration of the action description file 11 by the side of the server machine 1 consists of a part 312 which performs description about what does not use the default check approach among the part 311 which specifies the default actuation about the check of a version inequality, and subsequent actuation, and the program included in a certain product about the corresponding product. the version control approach by this invention -- a product/program -- management in which unit is possible.

[0038] Next, drawing 4 is a flow chart which shows the system behavior flow in the software version control method of the network system of the gestalt 1 of operation

concerning this invention. The version control server program 8 by the side of the server machine 1 always operates, in order to receive the version check request from a client machine 2 to the timing of arbitration, and it waits for the version check request from a client machine 2. Starting of a client program 13 is performed through the version control client program 12.

[0039] If the version control client program 12 of a client machine 2 is started, it will read the action description file 11 first (step S1). As a result of reading the action description file 11, when it is judged that the version check for application activation is required (step S2), the version control client program 12 sets up version information required for version checks, such as a version number of a self-program, an application name, and a product name, transmits version control server program 8 ** of the server machine 1 for the set-up version information, and performs a version check request (step S3).

[0040] If the version control server program 8 of the server machine 1 receives the version information transmitted from a client machine 2 and a version check request is received (step S4), it will read the action description file 9 on the server machine 1 (step S5). And according to the check approach described in the action description file 9, the version control server program 8 performs version check processing between a client program 13 and the server program 14 (step S6), and returns the version check result of whether a client program 13 and the server program 14 are the same versions to the version control client program 12 of the client machine 2 of version check request origin (step S7).

[0041] The version control client program 12 of a client machine 2 starts the application client program 13 succeedingly, when the version check result transmitted from the server machine 1 is received (step S8) and the notice of the version check result of the purport whose version of a client program and a server program is coincidence is received (step S9) (step S10).

[0042] On the other hand, when it is judged that the version of the version control client program 12 of a client machine 2 of a client program 13 and the server program 14 does not correspond (step S9), it judges whether software is replaced immediately (step S11). When it judges [replacing software immediately and] (step S11), the version control client program 12 of a client machine 2 replaces software according to the action description file 11 (step S12), and starts the application client program 13 (step S13).

[0043] The version control client program 12 of a client machine 2 judges whether a client program 13 is started, when it judges [not replacing software immediately and] (step S11) (step S14). When it judges [starting a client program 13 and] (step S14), the version control client program 12 of a client machine 2 starts a client program 13 (step S15), and directs the software exchange according to the action description file 11 (step S16).

[0044] Thus, with the gestalt of this operation, the version information of the client program 13 required for a version check is set up by the client machine 2 side. The set-up version information is transmitted to the server machine 1, and a version check request is performed. By the server machine 1 side Receive the version information of the client program 13 transmitted from a client machine 2, and a version check request is received. After performing the version check of the server program 14 read from the version information and the version control database 6 of the client program 13 which received, it constitutes so that the version check result may be transmitted to a client machine 2. Furthermore, when it is judged that it judges whether the version of a client program 13 and the server program 14 is in agreement based on the version check result transmitted from the server machine 1, and is not in agreement with the gestalt of this operation with a client machine 2 side, it constitutes so that the client program 13 which replaced the client program 13 whose version does not correspond so that it might be in agreement with the version of the server program 14, and replaced it may be performed. For this reason, since the version of the client program 13 to perform can be certainly doubled with the version of the server program 14 when performing a client program 13, system behavior which could be prevented from happening at the time of client program 13 activation, and was stabilized in the mismatching by the version inequality at it can be performed. Therefore, processing in the environment which always corresponded can be guaranteed.

[0045] Moreover, rather than the case where software is got and replaced from the conventional distribution server, since it constituted from a gestalt of this operation so that the client program 13 which was a version inequality in the client machine 2 side might be replaced, even if a network error arises, a client program 13 can be replaced certainly.

[0046] The gestalt of gestalt 2. book implementation of operation explains only the description part concerning invention of claim 2, and is explained with reference to drawing 1 and 4. Although the case where transmitted the version information set up immediately to the server machine 1, and a version check request was performed was explained with the gestalt 1 of the above-mentioned implementation after setting up the version information of a client program 13 by the client machine 2 side With the gestalt of this operation, the timing of a version check is specified as the action description file 11 of a client machine 2. By the version control client program 12 When it is judged that the timing of the specified version check was reached, it constitutes so that the version information of the set-up client program 13 may be transmitted to the server machine 1 and a version check request may be performed. In this case, since a version check can be performed to the specified timing, a version check can be performed efficiently. Hereafter, it explains using a drawing concretely.

[0047] Drawing 5 is drawing showing the example of description about the version

check timing setting in the action description file 11 shown in drawing 1 of the gestalt 2 of operation concerning this invention. What is necessary is just to check a version to the timing of starting of application in an application execution environment, in order to confirm certainly whether the version of software is in agreement between client/servers.

[0048] However, it always is not necessary to check a version depending on application in many cases. With the gestalt of this operation, it is characterized by carrying out according to the action description file 11 about the timing which carries out version check processing at the time of this application starting. Assignment [which / which are not carried / every program starting (every) 1 time / (1st), and / out out (none) / check] is performed in the action description file 11. [of the beginning on /the 1st]

[0049] It is the setup 502 which checks for every program starting about the application 2 (app.2) of this action description file 11, and is the setup 501 which checks only the first time of the beginning among days about application 1 (app.1). Actuation of the version control client program 13 in the condition that this setup was performed is explained based on the flow chart of drawing 4.

[0050] The version control client program 12 of a client machine 2 reads the action description file 11 at the time of the starting, and acquires the information on whether a version check is carried out from the Check field. Consequently, like [it is a setup of the action description file 11 of drawing 5 , and / when starting app.2], when it is judged that a version check needs to be carried out (step S2), the version control client program 12 transmits the version information of the set-up client program to the version control server program 8 of the server machine 1, and performs a version check request (step S3).

[0051] On the other hand, the version control client program 12 of a client machine 2 is a setup of drawing 5 , when it is judged that it is not necessary to perform a version check like [at the time of starting of the 2nd henceforth when starting app.1 several times continuously] (step S2), does not publish the version check request to the version control server program 8 of the server machine 1, but starts a client program 13 (step S10).

[0052] Thus, since a version check can be performed to the timing specified by specifying the timing of a version check as the action description file 11 of a client machine 1, and constituting from a gestalt of this operation so that the version information of the set-up client program may be transmitted to the server machine 1 and a version check request may be performed by the version control client program 12, when it is judged that the timing of the specified version check was reached, a version check can be performed efficiently. For example, implementation of a version check or download of software cannot be made to perform at every application starting. For this reason, since an unnecessary version check cannot be made to

perform on employment of a system, degradation of the engine performance and increase of an unprepared load can be prevented.

[0053] The gestalt of gestalt 3. book implementation of operation explains only the description part concerning invention of claim 3, and explains it with reference to drawing 1, and 4 and 6. With the gestalt 1 of the above-mentioned implementation, by the version control server program 8 If a version check request is received from a client machine 2 Although the case where compared the version information of the client program which received with the version information of the server program read from the version control database 6, and a version check was performed was explained With the gestalt of this operation, before performing a version check, the version check information that it uses for a version check is set as the action description file 9 of the server machine 1. By the version control server program 8 a version check is performed based on the version check information read from the action description file 9 (steps S5 and S6) -- it constitutes like. In this case, since a version check can be performed based on the set-up version check information, the flexible version interpretation approach doubled with software can be specified. Hereafter, it explains using a drawing concretely.

[0054] Drawing 6 is drawing showing the contents of the action description file 9 shown in drawing 1 of the gestalt 3 of operation concerning this invention. In case the version of software is checked, to the information which is large as information used as the criteria of a check, and serves as these check criteria the creation data ("date") of a file, a file name (in for example, the case of the product name containing a version), and the version information ("revision") embedded into the file or the module -- and The case where the version control of the contents of registration to the version control database 6 ("config"), for example, a product and an application group, is carried out collectively etc. is mentioned.

[0055] However, although there is much information used as the criteria of a version check, it is difficult to combine them and to perform a version check. The gestalt of this operation sets up the information used for a version check by describing any of the contents of registration ("config") of the version information ("revision") / version control database 6 embedded by a file creation date ("date") / software in the "Information" field 611,616 they are.

[0056] For example, in drawing 6, since "config" is described by the Information field about the product PRO 1, priority will be given to the version check based on the version control database 6. The version control server program 8 compares a version with this Information based on the information on description. If there is no difference in the contents of registration of the version control database 6 even if the version (revision) contained in the module of the case of the example of drawing 6, for example, a client program 13 and the server program 14, has a difference, the difference of a version will be judged to be what is not between a client program 13

and the server program 14.

[0057] Thus, before performing a version check with the gestalt of this operation, the version check information that it uses for a version check sets as the action description file 9 of the server machine 1, and since a version check can perform based on the version check information set up by constituting so that a version check may be performed based on this set-up version check information, the flexible version interpretation approach which doubled with software can specify by the version control server program 8. For this reason, since it can respond to the version information which changes with software, efficient systems operation is realizable.

[0058] The gestalt of gestalt 4. book implementation of operation explains only the description part concerning invention of claim 4, and explains it with reference to drawing 1, and 4 and 6. With the gestalt 1 of the above-mentioned implementation, by the version control server program 8 If a version check request is received from a client machine 2 Although the case where compared the version information of the client program which received with the version information of the server program read from the version control database 6, and a version check was performed was explained With the gestalt of this operation, before performing a version check, the check logic information at the time of a version check is set as the action description file 9. By the version control server program 8 a version check is performed based on the check logic information read from the action description file 9 (steps S5 and S6) -- it constitutes like. In this case, since a version check can be performed based on the set-up check logic information, the flexible version interpretation approach doubled with software can be specified.

[0059] The gestalt of this operation is related with the algorithm of a version check. As for the version of the client/server mold application distributed in the network environment in the actual system, it is desirable that it is in agreement about the same product. However, in actual system environment, the environment where software cannot be replaced easily also has the case of the 24-hour continuous-running implementation in a server machine etc. If it turns out that an actuation top problem does not occur although there is no exact match of a version in order to lessen exchange of software as much as possible when the versions of the application between client/servers have differed within such environments, it is desirable to regard it as version coincidence.

[0060] With the gestalt of this operation, a setup about the view (full coincidence or partial coincidence) of a version check is performed by establishing the "Algorithm" field 611,616 in the action description file 9 of the server machine 1, and describing any which are regarded as the version having been in agreement when coincidence of a version was completely the need ("complete") / version within the limits of specification between the client program 13 and the server program 14 ("range") they are.

[0061] Thus, the flexible version interpretation approach doubled with software can be specified by constituting from a gestalt of this operation so that the version control server program 8 may perform a version check based on the check logic information set as the action description file 9. For this reason, also in complicated environment-izing which cannot respond only by the simple version comparison of the environment where the software with which versions differ is intermingled etc., it can respond by customizing a version check algorithm.

[0062] The gestalt of gestalt 5. book implementation of operation explains only the description part concerning invention of claim 5, and explains it with reference to drawing 1, and 4 and 6. With the gestalt 1 of the above-mentioned implementation, by the version control server program 8 If a version check request is received from a client machine 2 Although the case where compared the version information of the client program 13 which received with the version information of the server program 14 read from the version control database 6, and a version check was performed was explained With the gestalt of this operation, before performing a version check, the dependency information between the software relevant to the action description file 9 is set up. By the version control server program 8 a version check is performed based on the dependency information between the related software read from the action description file 9 (steps S5 and S6) -- it constitutes like. In this case, since a version check can be performed based on the dependency information between the set-up related software, the version check of the whole software with a dependency can be put in block, and can be performed.

[0063] The gestalt of this operation is related with the version check approach of having taken the dependency of software into consideration. Within a distributed-system environment, the application in a dependency exists mutually mostly. For example, as application contained in a product 1, a client program client1 and the server program server2 exist, and the case where a client program client2 and the server program server2 exist is considered as application contained in a product 2.

[0064] Here, although the version between the client program client1 contained in a product 1 and the server program server1 is in agreement when only the application program contained in a product 2 is distributed in a distributed-system environment, the problem of being unable to perform a client program client1 or producing an error in an activation result may occur. Since the software in a product 1 is using the function in a product 2, this is generated.

[0065] As shown in drawing 6, with the gestalt of this operation, version coincidence is checked [by establishing the field "dependency" 613,618 in the action description file 9, and describing the dependency information between a product PRO 1 and a product PRO 2] as a check of the version at the time of starting by performing comparison processing not only with the product PRO 1 but the version in a product PRO 2 at the time of the demand issue to SERVER1 in a product PRO 1.

[0066] Thus, since a mutual dependency can be specified to the software which straddles a management unit and has a dependency by constituting from a gestalt of this operation so that the version control server program 8 may perform a version check based on the dependency information between the related software set as the action description file 9, efficient systems operation is realizable. With the gestalt of this operation, since the software and the software in a dependency can be replaced in case one software is replaced, a system administrator's burden is mitigable.

[0067] The gestalt of gestalt 6. book implementation of operation explains only the description part concerning invention of claim 6, and explains it with reference to drawing 1, and 4 and 7. With the gestalt 1 of the above-mentioned implementation, by the version control client program 12 Although the case where software was replaced was explained when the version of a client program 13 and the server program 14 was not in agreement (step S9) and it judged [replacing software immediately and] (step S11) With the gestalt of this operation, before replacing a client program 13 The timing which replaces a client program 13 is set as the action description file 11. By the version control client program 12 When the exchange timing of the set-up client program 13 is reached (step S11), it constitutes so that a client program 13 may be replaced. In this case, since a client program 13 can be replaced when the exchange timing of the set-up client program 13 is reached, the software exchange doubled with systems operation can be carried out.

[0068] The gestalt of this operation is related with the software exchange approach when a difference occurs in the version information of a client program 13, and the version information of the server program 14 as a result of version check implementation. When it becomes clear that it is not a right version for operating application in a distributed-system environment as a result of a version check, it is desirable to usually replace application. Here, especially when a system is working in there being much number of a client machine 2, the exchange activity of this application is difficult and it is desirable to cancel a version mismatch by the software exchange approach according to system environment.

[0069] The gestalt of this operation is related with an approach to change the software used as the version inequality at the time of failure to the version check according to the contents of description of the action description file 11. Description of this action description file 11 is equivalent to 212 parts of drawing 2. Drawing 7 is drawing showing the concrete example of description of 212 parts shown in drawing 2. When the gestalt of this operation is judged that exchange of software is required as a result of a version check, it establishes a means to specify the timing which does the exchange activity of an actual program. as exchange timing, immediately after the inequality detection ("now") by the version check and after inequality detection operate, and application depends them on time-of-day assignment immediately after the own termination ("after") of application -- replacing (exchange of "time" and

operating end time etc.) -- it can set up "now", "after", and "time" -- which information is described in the "When" field of the action description file 11 of the server machine 1.

[0070] When a version inequality is detected about program app-a in the example of description of the action description file 11 of drawing 7, it is a setup for not performing actual processing of application but performing exchange 711 immediately, and is a setup for replacing time amount (outside of operating time amount) about program app-b at midnight (717). Actuation is explained drawing 4 and based on 7.

[0071] When the version control client program 12 of a client machine 2 is started and detects the inequality of a version check, it is replaced by processing of the part of step 9 of drawing 4, and judges conditions. And as description of the action description file 11, immediately, when [to be changed] it belongs to product PRO-X of drawing 7 (step S11), to a distribution program, the version control client program 12 notifies the conditions of required software, and replaces software of a client program 13 (step S12).

[0072] The version control client program 12 of a client machine 2 starts the application of the replaced client program 13, after checking an exchange environment (step S13). On the other hand, required application distribution processing is registered to the schedule function in which the version control client program 12 carries out the system distribution of the version control client program 12 when exchange operating time amount outside to which it belongs to product PRO-Y of drawing 7 as description of the action description file 11 is specified etc. (step 16). As contents of registration, assignment of distribution time amount and the product for distribution is mentioned.

[0073] Thus, with the gestalt of this operation, before replacing a client program 13 The timing which replaces a client program 13 is set as the action description file 11. By the version control client program 12 Since it constituted so that a client program 13 might be replaced when the exchange timing of the set-up client program 13 was reached (step S11), Since a client program 13 can be replaced when the exchange timing of the set-up client program 13 is reached, the software exchange doubled with systems operation can be carried out.

[0074] With the gestalt of this operation, although he wants to replace software immediately now, when not being changed for example, it can specify beforehand changing at night, or changing, after a series of business is completed. For this reason, since the exchange schedule of software can be flexibly set up according to the requirements for a system, efficient systems operation is realizable.

[0075] The gestalt of gestalt 7. book implementation of operation explains only the description part concerning invention of claim 7, and explains it with reference to drawing 1, and 4 and 7. With the gestalt 1 of the above-mentioned implementation, by the version control client program 12 Although the case where software was replaced

was explained when the version of a client program 13 and the server program 14 was not in agreement (step S9) and it judged [replacing software immediately and] (step S11) With the gestalt of this operation, before replacing a client program 13 The exchange conditions of a client program 13 are set as the action description file 11 of a client 2. By the version control client program 12 It constitutes so that a client program 13 may be replaced based on the exchange conditions of the set-up client program 13. In this case, since a client program 13 can be replaced based on the exchange conditions of the set-up client program 13, what has a high significance can realize the flexible software exchange according to the environment and the work breakdown of performing a version check frequently, for example.

[0076] When the gestalt of this operation is judged that a client program 13 needs to be changed as a result of a version check, it changes according to the exchange conditions of the client program 13 set up in advance. In case software is replaced, software used as the candidate for exchange cannot usually carry out by operating state. Then, when it is [operating] under operation in a real system, it may be difficult to do this exchange activity in time. It is not concerned under operating operation or with ****, but exchange is the need immediately in many cases. Conditions, such as a terminal of the software with which it is used frequently, for example, and specification which does an important activity, and very important specific business, are applied.

[0077] With the gestalt of this operation, the "Whether" fields 712 and 717 are established in the action description file 11 of a client machine 2, and by the version check response from the version control server program 8 of the server machine 1, also when a version receives a notice called an inequality, according to the contents of the "Whether" field of the action description file 11, the version control client program 12 of a client machine 2 carries out exchange processing of the inharmonious client program 12 (steps S11 and S12). For example, in drawing 7 , when the version difference more than N time is detected about the same application by describing the information on the count (times) of a version inequality about product PRO-X, software of a client program 13 is replaced.

[0078] Thus, with the gestalt of this operation, before replacing a client program 13, a client program 13 can be replaced based on the exchange conditions of the client program 13 set up by setting the exchange conditions of a client program 13 as the action description file 11 of a client 2 since it constituted so that a client program 13 might be replaced based on the exchange conditions of the client program 13 set up by the version control client program 12. For this reason, for example, what has a high significance can realize the flexible software exchange according to the environment and the work breakdown of performing a version check frequently.

[0079] The gestalt of gestalt 8. book implementation of operation explains only the description part concerning invention of claim 8, and explains it with reference to

drawing 1 , and 4 and 7. With the gestalt 1 of the above-mentioned implementation, by the version control client program 12 Although the case (step S12) where software was replaced was explained when the version of a client program 13 and the server program 14 was not in agreement (step S9) and it judged [replacing software immediately and] (step S11) When the gestalt of this operation is judged that a client program 13 needs to be changed as a result of a version check, it establishes a means to choose the client program 13 for exchange. When the difference of the version of a client program 13 is detected by each client machine 2 within a distributed-system environment and it is judged that a client program 13 needs to be changed, in order to suppress the effect of the business on [under operation accompanying the exchange activity of a client program 13] to the minimum, it is desirable to replace collectively not all the software groups that constitute a certain specific business, but to replace only a necessary minimum application program.

[0080] With the gestalt of this operation, when the field 713,718 is formed and the inequality of a product is detected between machines, only the module (file of execute form) described in the "What" "What" field out of the corresponding product is changed to the action description file 11. For example, in drawing 7 , in case it presupposed that product PRO-X consisted of 100 modules and exchange of the client program 13 by the version inequality is needed, all 100 modules are not replaced but exchange of three modules "prog_1" of description, "prog_2", and "prog_3" is carried out in the "What" field.

[0081] Thus, the conditions which change to the action description file 11 of a client machine 2, and choose the target client program 13 are set up, and before replacing a client program 13, it constitutes from a gestalt of this operation so that a client program 13 may be replaced based on the client program 13 for [which was set up] exchange by the version control client program 12. In this case, since it can change and a client program 13 can be replaced based on the target client program 13, the set-up related client program 13 cannot be replaced collectively, but it can change per module. Therefore, since exchange in the module unit doubled with a system or the requirements for operating can be performed, the efficient systems operation which does not replace an unnecessary module as much as possible is realizable.

[0082]

[Effect of the Invention] According to invention according to claim 1, the version information of a client program required for a version check is set up by the client side. The set-up version information is transmitted to a server and a version check request is performed. By the server side Receive the version information of the client program transmitted from a client, and a version check request is received. After performing the version check of the server program read from the version information and the version control database of the client program which received, It constitutes so that the version check result may be transmitted to a client machine. Further by the client

side Based on the version check result transmitted from a server, it judges whether the version of a client program and a server program is in agreement. Since it constituted so that the client program which replaced the client program whose version does not correspond so that it might be in agreement with the version of a server program, and replaced it might be performed when it was judged that it is not in agreement, Since the version of the client program to perform can be certainly doubled with the version of a server program when performing a client program System behavior which could be prevented from happening at the time of client program activation, and was stabilized in the mismatching by the version inequality at it can be performed. Therefore, processing in the environment which always corresponded can be guaranteed.

[0083] Moreover, rather than the case where software is got and replaced from the conventional distribution server, since according to invention according to claim 1 it constituted so that the client program which was a version inequality in the client side might be replaced, even if a network error arises, a client program can be replaced certainly.

[0084] Since a version check can be performed to the timing specified by constituting so that the version information of the set-up client program may be transmitted to a server and a version check request may be performed when it is judged according to invention according to claim 2 that the timing of the version check which specifies the timing of a version check as the client and was specified was reached, a version check can be performed efficiently. For example, implementation of a version check or download of software cannot be made to perform at every application starting. For this reason, since an unnecessary version check cannot be made to perform on employment of a system, degradation of the engine performance and increase of an unprepared load can be prevented.

[0085] Since according to invention according to claim 3 a version check can be performed based on the version check information set up by setting the version check information that it uses for a version check as the server, and constituting so that a version check may be performed based on this set-up version check information before performing a version check, the flexible version interpretation approach doubled with software can be specified. For this reason, since it can respond to the version information which changes with software, efficient systems operation is realizable.

[0086] According to invention according to claim 4, the flexible version interpretation approach doubled with software can be specified by constituting so that a version check may be performed based on the check logic information set as the server. For this reason, also in complicated environment-izing which cannot respond only by the simple version comparison of the environment where the software with which versions differ is intermingled etc., it can respond by customizing a version check algorithm.

[0087] Since a mutual dependency can be specified to the software which straddles a management unit and has a dependency by constituting so that a version check may be performed based on the dependency information between the related software set as the server according to invention according to claim 5, efficient systems operation is realizable. For example, since the software and the software in a dependency can be replaced in case one software is replaced, a system administrator's burden is mitigable.

[0088] Since it constituted so that a client program might be replaced when according to invention according to claim 6 the exchange timing of the client program which sets the timing which replaces a client program as the client, and was set up was reached before replacing a client program and a client program can be replaced when the exchange timing of the set-up client program is reached, the software exchange doubled with systems operation can be carried out.

[0089] According to invention according to claim 7, since it constituted so that a client program might be replaced based on the exchange conditions of the client program which sets the exchange conditions of a client program as the client, and was set up before replacing a client program, a client program can be replaced based on the exchange conditions of the set-up client program. For this reason, for example, what has a high significance can realize the flexible software exchange according to the environment and the work breakdown of performing a version check frequently.

[0090] Before replacing a client program according to invention according to claim 8, since it constituted so that it might change and a client program might be replaced based on the target client program, and it can change and a client program can be replaced based on the target client program, the set-up related client program cannot be replaced collectively, but it can change [which sets up the conditions which change to a client and choose the target client program, and was set up] per module. Therefore, since exchange in the module unit doubled with a system or the requirements for operating can be performed, the efficient systems operation which does not replace an unnecessary module as much as possible is realizable.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]

[Drawing 1] It is the block diagram showing the system configuration of the software version control method of the network system of the gestalt 1 of operation concerning this invention.

[Drawing 2] It is drawing showing the configuration of the action description file on the client machine shown in drawing 1.

[Drawing 3] It is drawing showing the configuration of the action description file on the server machine shown in drawing 1 .

[Drawing 4] It is the flow chart which shows the system behavior flow in the software version control method of the network system of the gestalt 1 of operation concerning this invention.

[Drawing 5] It is drawing showing the example of description about the version check timing setting in the action description file of the gestalt 2 of operation concerning this invention.

[Drawing 6] It is drawing showing the contents of the action description file of the gestalt 3 of operation concerning this invention.

[Drawing 7] It is drawing showing the contents of the action description file of the gestalt 6 of operation concerning this invention.

[Drawing 8] It is the flow chart which shows the system behavior flow in the version control method of the conventional software.

[Drawing 9] It is the flow chart which shows the system behavior flow in the version control method of the conventional software.

[Description of Notations]

1 A server machine, 2 A client machine, 3 Communication media, 4 5 A software package, 7 A magnetic disk drive, 6 A version control database, 8 9 A version control server program, 11 An action description file, 12 A version control client program, 13 A client program, 14 16 A server program, 17 Distribution program.

(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平10-21059

(43)公開日 平成10年(1998)1月23日

(51)Int.Cl. ⁶	識別記号	序内整理番号	F I	技術表示箇所
G 06 F 9/06	4 1 0		G 06 F 9/06	4 1 0 Q
13/00	3 5 1		13/00	3 5 1 H
	3 5 7			3 5 7 Z

審査請求 有 請求項の数 8 O.L (全 17 頁)

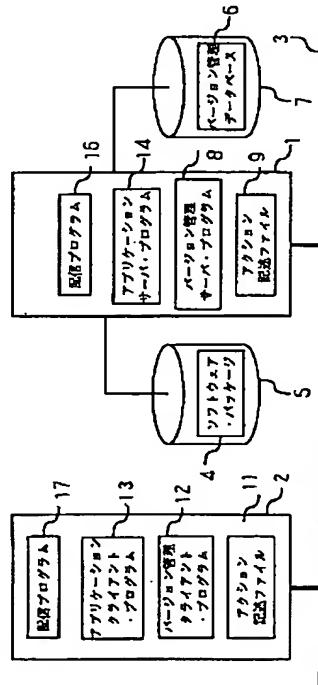
(21)出願番号	特願平8-172271	(71)出願人	000006013 三菱電機株式会社 東京都千代田区丸の内二丁目2番3号
(22)出願日	平成8年(1996)7月2日	(72)発明者	百本 征弘 東京都千代田区丸の内二丁目2番3号 三菱電機株式会社内
		(72)発明者	金森 阜郎 東京都千代田区丸の内二丁目2番3号 三菱電機株式会社内
		(74)代理人	弁理士 宮田 金雄 (外3名)

(54)【発明の名称】 ネットワークシステムのソフトウェア・バージョン管理方式

(57)【要約】

【課題】 クライアントのソフトウェアを実行する時、実行するクライアントのソフトウェアのバージョンを確実に合わせて、バージョン不一致による不整合を起こらないようにして安定したシステム動作を実現することができるネットワークシステムのソフトウェア・バージョン管理方式を得る。

【解決手段】 クライアント1は、クライアント・プログラム13とサーバ・プログラム14のバージョンが一致していないと判断すると、バージョンが一致していないクライアント・プログラム13をサーバ・プログラム14のバージョンと一致させるに入れ替え、入れ替えたクライアント・プログラム13を実行する。



1

【特許請求の範囲】

【請求項1】 クライアントとサーバがネットワークを介して接続されたネットワークシステムのソフトウェア・バージョン管理方式において、バージョンチェックに必要なクライアント・プログラムのバージョン情報を設定するバージョン情報設定手段と、設定したバージョン情報をサーバへ転送してクライアント・プログラムのバージョンチェック要求を行なうバージョンチェック要求手段とをクライアントに設け、クライアントから転送されるクライアント・プログラムのバージョン情報を受信して、バージョンチェック要求を受け付けるバージョンチェック要求受け付け手段と、サーバ・プログラムのバージョン情報を格納するバージョン情報格納手段と、受信したクライアント・プログラムのバージョン情報とバージョン情報格納手段から読み出したサーバ・プログラムのバージョン情報を比較して、クライアント・プログラムとサーバ・プログラムのバージョンチェックを行なうバージョンチェック手段と、バージョンチェック結果をクライアントへ転送するバージョンチェック結果転送手段とをサーバに設け、サーバから転送されるバージョンチェック結果に基づいて、クライアント・プログラムのバージョンがサーバ・プログラムのバージョンと一致していないかを判断するバージョン不一致判断手段と、バージョンが一致していないと判断した場合、バージョンが一致していないクライアント・プログラムをサーバ・プログラムのバージョンと一致するように入れ替えるプログラム入れ替え手段と、入れ替えたクライアント・プログラムを実行するプログラム実行手段とをクライアントに設けることを特徴とするネットワークシステムのソフトウェア・バージョン管理方式。

【請求項2】 バージョンチェック要求を行なう前に、バージョンチェックのタイミングを指定するタイミング指定手段と、指定したバージョンチェックのタイミングに達したかを判断するタイミング判断手段とをクライアントに設け、前記バージョンチェック要求手段は、指定したバージョンチェックのタイミングに達したと判断した場合、設定したクライアント・プログラムのバージョン情報をサーバへ転送してバージョンチェック要求を行なうことを特徴とする請求項1に記載のネットワークシステムのソフトウェア・バージョン管理方式。

【請求項3】 バージョンチェックを行なう前に、バージョンチェックに用いるバージョンチェック情報を設定するチェック情報設定手段をサーバに設け、前記バージョンチェック手段は、設定したバージョンチェック情報に基づいてバージョンチェックを行なうことを特徴とする請求項1、2の何れかに記載のネットワークシステムのソフトウェア・バージョン管理方式。

【請求項4】 バージョンチェックを行なう前に、バージョンチェック時のチェックロジックを設定するロジッ

2

ク設定手段をサーバに設け、前記バージョンチェック手段は、設定したチェックロジックに基づいてバージョンチェックを行なうことを特徴とする請求項1、2の何れかに記載のネットワークシステムのソフトウェア・バージョン管理方式。

【請求項5】 バージョンチェックを行なう前に、関連するソフトウェア間の依存関係を設定する依存関係設定手段をサーバに設け、前記バージョンチェック手段は、設定した関連するソフトウェア間の依存関係に基づいてバージョンチェックを行なうことを特徴とする請求項1、2の何れかに記載のネットワークシステムのソフトウェア・バージョン管理方式。

【請求項6】 クライアント・プログラムを入れ替える前に、クライアント・プログラムを入れ替えるタイミングを設定するタイミング設定手段と、設定したクライアント・プログラムの入れ替えタイミングに達したかを判断するタイミング判断手段とをクライアントに設け、前記プログラム入れ替え手段は、設定したクライアント・プログラムの入れ替えタイミングに達したと判断した場合、クライアント・プログラムを入れ替えることを特徴とする請求項1～5の何れかに記載のネットワークシステムのソフトウェア・バージョン管理方式。

【請求項7】 クライアント・プログラムを入れ替える前に、クライアント・プログラムの入れ替え条件を設定する入れ替え条件設定手段をクライアントに設け、前記プログラム入れ替え手段は、設定したクライアント・プログラムの入れ替え条件に基づいてクライアント・プログラムを入れ替えることを特徴とする請求項1～5の何れかに記載のネットワークシステムのソフトウェア・バージョン管理方式。

【請求項8】 クライアント・プログラムを入れ替える前に、入れ替え対象のクライアント・プログラムを選択する入れ替え対象選択手段をクライアントに設け、前記プログラム入れ替え手段は、選択した入れ替え対象のクライアント・プログラムを入れ替えることを特徴とする請求項1～5の何れかに記載のネットワークシステムのソフトウェア・バージョン管理方式。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ネットワークシステムのソフトウェア・バージョン管理方式に係り、詳しくは、分散システム上で動作するソフトウェア・バージョン管理方式に適用することができ、特に、クライアントのソフトウェアを実行する時、実行するクライアントのソフトウェアのバージョンを確実に合わせて、バージョン不一致による不整合を起こらないようにして安定したシステム動作を実現することができるネットワークシステムのソフトウェア・バージョン管理方式に関する。

【0002】

【従来の技術】分散システム上で動作する従来のソフト

3

ウェアのバージョン管理方式では、バージョン管理用のデータベースにより各マシンにインストールされているソフトウェアおよびそのバージョン情報を集中的に管理している。この従来のソフトウェアのバージョン管理方式には、該ソフトウェア及びそのバージョン情報を管理した後、スケジュールを設定した時のソフトウェア配布機能やソフトウェア配布に失敗した際のリトライ機能などにより、ソフトウェア配信機能の付加機能として配信時に各クライアントのソフトウェアのバージョンの整合性を保証しようという形態のものや、分散システム全体のバージョン情報を管理して、定期的に各クライアントのソフトウェアのバージョンのチェックを行ない、バージョンの不一致を検出した際に、バージョンを合わせるために、バージョンの合っていないクライアントにソフトウェアの入れ替えを指示する形態のものが知られている。

【0003】まず、前者のスケジュール機能及びソフトウェア配布に失敗した際のリトライ機能によりシステム全体のバージョンの一一致を保証しようという形態の従来のソフトウェアのバージョン管理方式について、以下に図面を用いて説明する。図8は従来のソフトウェアのバージョン管理方式におけるシステム動作フローを示すフローチャートである。ソフトウェア配布サーバは、配布するソフトウェアの配布先、入れ替え時間などのスケジュール情報を設定した後（ステップS801）、設定したスケジュール情報に従って、配布先のクライアントに対するソフトウェアの配信処理を行ない、クライアントへソフトウェア入れ替え用ファイルを配信する（ステップS802）。この時、設定したスケジュール情報のソフトウェアの入れ替え時間などの情報もクライアントへ送信される。

【0004】クライアントは、ソフトウェア配布サーバから配信されたソフトウェア入れ替え用ファイルを受信すると（ステップS803）、ソフトウェア入れ替え用ファイルの受信結果をソフトウェア配布サーバへ通知する（ステップS804）。クライアントは、ソフトウェア配布サーバから配信されたソフトウェア入れ替え用ファイルを受信し、ソフトウェア入れ替え用ファイルの受信が正常に行なわれた場合、ソフトウェア配布サーバから送信され受信したスケジュール情報のソフトウェアの入れ替え時間などの情報に従って、ソフトウェアの入れ替え処理を実施する（ステップS805）。

【0005】クライアントは、ソフトウェア入れ替え処理の結果をソフトウェア配布サーバへ通知する（ステップS806）。ソフトウェア配布サーバは、クライアントから通知されたソフトウェア入れ替え用ファイル受信確認結果とソフトウェア入れ替え処理結果のソフトウェア配布結果を受信し（ステップS807）、受信したソフトウェア配布結果を基にソフトウェア配布が正常に処理されたか否かを判断する（ステップS808）。

4

【0006】ソフトウェア配布サーバは、ソフトウェア配布が正常に処理されず失敗していると判断した場合（ステップS808）、設定したスケジュール情報を変更せずに、再度クライアントへソフトウェアを配信する（ステップS802）。一方、ソフトウェア配布サーバは、ソフトウェア配布が正常に処理されたと判断した場合（ステップS808）、他に入れ替え対象ソフトウェアがあるか否かを判断し（ステップS809）、ある場合は、次のソフトウェアの処理に戻り（ステップS8010）、ない場合は、処理を終了する。

【0007】この従来のソフトウェアのバージョン管理方式では、配布サーバ側で設定したスケジュール情報に従ってクライアントが定期的にソフトウェアを入れ替えたり、配布サーバからのソフトウェア配布を失敗した時やクライアントがソフトウェアの入れ替えを失敗した時、再度配布サーバからソフトウェアを配信してソフトウェアの入れ替え処理を行なうものである。

【0008】次に、前述した後者の定期的にバージョンのチェックを行なってバージョンの不一致を検出した際

20 に、ソフトウェアの入れ替えを指示する形態の従来のソフトウェアのバージョン管理方式について、以下に図面を用いて説明する。図9は従来のソフトウェアのバージョン管理方式におけるシステム動作フローを示すフローチャートである。クライアントは、定期的にバージョンチェックに必要な情報を設定し、設定した情報を基に定期的にソフトウェアのバージョンチェックをバージョン管理サーバへ依頼する（ステップS901）。この時、クライアントは、クライアントのソフトウェアのバージョン情報やソフトウェアのバージョンチェック要求情報をなどをバージョン管理サーバへ送信する。

30 【0009】バージョン管理サーバは、クライアントからソフトウェアのバージョンチェック要求を受け付けると（ステップS902）、バージョン管理データベースに管理しているバージョン情報を基に、クライアントのソフトウェアのバージョン情報をチェックして、バージョンの整合性を確認する（ステップS903）。バージョン管理サーバは、バージョン・チェックの結果情報をクライアントへ送信して応答する（ステップS904）。

40 【0010】クライアントは、バージョン管理サーバから送信されるバージョン・チェック結果情報を受信すると（ステップS905）、受信したバージョン・チェック結果情報を基に、クライアントのソフトウェアのバージョンがバージョン管理サーバで管理されているソフトウェアのバージョンと一致しているか否か、即ち、ソフトウェアの入れ替えが必要であるか否かを判断する（ステップS906）。

50 【0011】クライアントは、クライアントのソフトウェアのバージョンがバージョン管理サーバで管理されているバージョンと一致して、クライアントのソフトウェ

5

アの入れ替えが必要でないと判断すると（ステップS906）、そのまま処理を終了する。一方、クライアントは、クライアントのソフトウェアのバージョンがバージョン管理サーバで管理されているバージョンと一致せず、クライアントのソフトウェアの入れ替えが必要であると判断すると（ステップS906）、ソフトウェア入れ替え用ファイルの配信をソフトウェア配布サーバへ要求する（ステップS907）。

【0012】ソフトウェア配布サーバは、クライアントからのソフトウェア配信要求に従ったスケジュール情報を設定し（ステップS908）、設定したスケジュール情報を基に、配信要求元のクライアントへソフトウェア入れ替え用ファイルを配信する（ステップS909）。この時、設定したスケジュール情報のソフトウェアの入れ替え時間などの情報もクライアントへ送信される。

【0013】クライアントは、ソフトウェア配布サーバから配信されたソフトウェア入れ替え用ファイルを受信すると（ステップS910）、受信したスケジュール情報の入れ替え時間などの情報に従って、ソフトウェア入れ替え処理を実施する（ステップS911）。

【0014】この従来のソフトウェアのバージョン管理方式では、定期的にクライアント側からバージョン管理サーバへソフトウェアのバージョンチェックを要求し、クライアントのバージョンがバージョン管理サーバのバージョンと一致していないと、クライアントがソフトウェア配布サーバから入れ替え処理用のソフトウェアを貰ってソフトウェアの入れ替え処理を行なうものである。

【0015】

【発明が解決しようとする課題】上記した前者の従来のソフトウェアのバージョン管理方式では、配布サーバ側で設定したスケジュール情報に従ってクライアントが定期的にソフトウェアを入れ替えることができるとともに、配布サーバからのソフトウェア配布を失敗した時やクライアントがソフトウェアの入れ替えを失敗した時、再度配布サーバからソフトウェアを配信してソフトウェアの入れ替え処理を行なうことができるが、クライアント側へのソフトウェア配布完了前にクライアント・プログラムが実行された際、サーバ・プログラムとの間でのバージョン一致は保証されておらず、また、バージョン不一致検出時も、クライアントのソフトウェアの入れ替え実施は、クライアント・プログラム起動のタイミングでは行なえない。このため、ソフトウェア実行時にバージョン不一致による不整合が生じて、安定したシステム動作を行ない難いという問題があった。

【0016】また、この前者の従来のソフトウェアのバージョン管理方式では、ネットワークエラー等が生じた時、配布サーバから入れ替え用のソフトウェアを貰えなくなることがあるため、ソフトウェアの入れ替えを行なえなくなることがあるという問題があった。

【0017】上記した後者のソフトウェアのバージョン

6

管理方式では、定期的にクライアント側からバージョン管理サーバへソフトウェアのバージョンチェックを要求し、クライアントのバージョンがバージョン管理サーバのバージョンと一致していない時、クライアントがソフトウェア配布サーバから入れ替え処理用のソフトウェアを貰ってソフトウェアの入れ替え処理を行なうことができるが、定期的にクライアントのソフトウェアのバージョンチェックを行なっているのであって、クライアントのソフトウェアを実行する時に、ソフトウェアの入れ替えを必ずしも行なっているのではないため、クライアントのソフトウェアを実行する時に、常にバージョンを一致させ難い。このため、ソフトウェア実行時にバージョン不一致による不整合が生じて、安定したシステム動作を行ない難いという問題があった。

【0018】また、この後者のソフトウェアのバージョン管理方式では、バージョンチェックの方式についても単純な一致検索を行なっており、異なるバージョン間で互換性の保証されたソフトウェアの入れ替えや、複数のバージョンの異なるクライアントアプリケーションとサーバとの複雑な関係に対応することができていなかつた。

【0019】また、この後者のソフトウェアのバージョン管理方式では、ネットワークエラー等が生じた時、配布サーバから入れ替え用のソフトウェアを貰えなくなることがあるため、ソフトウェアの入れ替えを行なえなくなることがあるという問題があった。

【0020】そこで、本発明は、クライアントのソフトウェアを実行する時、実行するクライアントのソフトウェアのバージョンを確実に合わせて、バージョン不一致による不整合を起こらないようにして安定したシステム動作を実現することができるとともに、ネットワークエラーが生じても、バージョン不一致だったクライアントのソフトウェアを確実に入れ替えることができるネットワークシステムのソフトウェア・バージョン管理方式を提供することを目的とする。

【0021】

【課題を解決するための手段】請求項1記載の発明は、クライアントとサーバがネットワークを介して接続されたネットワークシステムのソフトウェア・バージョン管理方式において、バージョンチェックに必要なクライアント・プログラムのバージョン情報を設定するバージョン情報設定手段と、設定したバージョン情報をサーバへ転送してクライアント・プログラムのバージョンチェック要求を行なうバージョンチェック要求手段とをクライアントに設け、クライアントから転送されるクライアント・プログラムのバージョン情報を受信して、バージョンチェック要求を受け付けるバージョンチェック要求受け付け手段と、サーバ・プログラムのバージョン情報を格納するバージョン情報格納手段と、受信したクライアント・プログラムのバージョン情報とバージョン情報格

7

納手段から読み出したサーバ・プログラムのバージョン情報を比較して、クライアント・プログラムとサーバ・プログラムのバージョンチェックを行なうバージョンチェック手段と、バージョンチェック結果をクライアントへ転送するバージョンチェック結果転送手段とをサーバに設け、サーバから転送されるバージョンチェック結果に基づいて、クライアント・プログラムのバージョンがサーバ・プログラムのバージョンと一致していないかを判断するバージョン不一致判断手段と、バージョンが一致していないと判断した場合、バージョンが一致していないクライアント・プログラムをサーバ・プログラムのバージョンと一致するように入れ替えるプログラム入れ替え手段と、入れ替えたクライアント・プログラムを実行するプログラム実行手段とをクライアントに設けることを特徴とするものである。

【0022】請求項2記載の発明は、上記請求項1記載の発明において、バージョンチェック要求を行なう前に、バージョンチェックのタイミングを指定するタイミング指定手段と、指定したバージョンチェックのタイミングに達したかを判断するタイミング判断手段とをクライアントに設け、前記バージョンチェック要求手段が、指定したバージョンチェックのタイミングに達したと判断した場合、設定したクライアント・プログラムのバージョン情報をサーバへ転送してバージョンチェック要求を行なうことを特徴とするものである。

【0023】請求項3記載の発明は、上記請求項1、2の何れかに記載の発明において、バージョンチェックを行なう前に、バージョンチェック用いるバージョンチェック情報を設定するチェック情報設定手段をサーバに設け、前記バージョンチェック手段が、設定したバージョンチェック情報に基づいてバージョンチェックを行なうことを特徴とするものである。

【0024】請求項4記載の発明は、上記請求項1、2の何れかに記載の発明において、バージョンチェックを行なう前に、バージョンチェック時のチェックロジックを設定するロジック設定手段をサーバに設け、前記バージョンチェック手段が、設定したチェックロジックに基づいてバージョンチェックを行なうことを特徴とするものである。

【0025】請求項5記載の発明は、上記請求項1、2記載の何れかに記載の発明において、バージョンチェックを行なう前に、関連するソフトウェア間の依存関係を設定する依存関係設定手段をサーバに設け、前記バージョンチェック手段が、設定した関連するソフトウェア間の依存関係に基づいてバージョンチェックを行なうことを特徴とするものである。

【0026】請求項6記載の発明は、上記請求項1～5の何れかに記載の発明において、クライアント・プログラムを入れ替える前に、クライアント・プログラムを入れ替えるタイミングを設定するタイミング設定手段と、

8

設定したクライアント・プログラムの入れ替えタイミングに達したかを判断するタイミング判断手段とをクライアントに設け、前記プログラム入れ替え手段が、設定したクライアント・プログラムの入れ替えタイミングに達したと判断した場合、クライアント・プログラムを入れ替えることを特徴とするものである。

【0027】請求項7記載の発明は、上記請求項1～5の何れかに記載の発明において、クライアント・プログラムを入れ替える前に、クライアント・プログラムの入れ替え条件を設定する入れ替え条件設定手段をクライアントに設け、前記プログラム入れ替え手段が、設定したクライアント・プログラムの入れ替え条件に基づいてクライアント・プログラムを入れ替えることを特徴とするものである。

【0028】請求項8記載の発明は、上記請求項1～5の何れかに記載の発明において、クライアント・プログラムを入れ替える前に、クライアント・プログラムの入れ替え対象を選択する入れ替え対象選択手段をクライアントに設け、前記プログラム入れ替え手段が、選択したクライアント・プログラムの入れ替え対象に基づいてクライアント・プログラムを入れ替えることを特徴とするものである。

【0029】

【発明の実施の形態】以下、本発明の実施の形態を図面を参照して説明する。

実施の形態1．図1は本発明に係る実施の形態1のネットワークシステムのソフトウェア・バージョン管理方式のシステム構成を示すブロック図である。ネットワークシステムのマシン構成は、任意の台数のサーバ・マシン1及びクライアント・マシン2からなり、各コンピュータは、LAN等の通信媒体3を介してネットワークに接続されている。サーバ・マシン1は、アプリケーション・プログラムのまとまりであるソフトウェア・パッケージ4を格納するための磁気ディスク装置5と、バージョン情報を管理するバージョン管理データベース6を格納するための磁気ディスク装置7とを有する。

【0030】次に、ソフトウェア構成について説明する。まず、サーバ・マシン1側のバージョン管理に関するソフトウェアは、各ネットワーク上の各ソフトウェアのバージョン情報を記録するバージョン管理データベース6と、クライアント・マシン2によるバージョン・チェック要求を受け付けるバージョン管理サーバ・プログラム8と、バージョン・チェック方法に関して規定したアクション記述ファイル9とから構成される。

【0031】クライアント・マシン2側のバージョン管理に関するソフトウェアは、バージョン管理サーバ・プログラム8に対するバージョン・チェック要求前後のソフトウェアの動作を規定するアクション記述ファイル11と、バージョン・チェック要求を発行するバージョン管理クライアント・プログラム12とから構成される。

9

【0032】分散システム環境下で動作するアプリケーションは、アプリケーションクライアント・プログラム13及びアプリケーションサーバ・プログラム14である。クライアント・プログラム13の起動は、バージョン管理クライアント・プログラム12を通して行なわれる。ソフトウェアの配布及びバージョンチェック後のソフトウェアの入れ替えは、サーバ・マシン1側の配信プログラム16がソフトウェア・パッケージ4を、クライアント・マシン2側の配信プログラム217に送ることにより行なわれる。

【0033】次に、アクション記述ファイルの構成について説明する。図2は図1に示すクライアント・マシン2上のアクション記述ファイル11の構成を示す図である。クライアント・マシン2上のアクション記述ファイル11には、バージョン・チェックを行なうタイミングに関する記述211と、バージョン・チェックによりクライアント・プログラムとサーバ・プログラムの間でバージョンが異なっていることが判った後のソフトウェア入れ替え方法に関する記述212を行なう。ソフトウェア入れ替え方法としては、ソフトウェアをいつ入れ替えるか、どのソフトウェアを入れ替えるか等の情報が定義されている。

【0034】次に、図3は図1に示すサーバ・マシン1上のアクション記述ファイル9の構成を示す図である。サーバ・マシン1上のアクション記述ファイル9には、ソフトウェアのバージョン・チェック方法に関する記述411を行なう。クライアント側/サーバ側何れのアクション記述ファイル9、11とも、“Product”221、321及び“Program”222、322というタグ名を有する。

【0035】“Program”は、個々のアプリケーション・プログラムを意味し、“Program”に引き続き該当するプログラム名を記述する。“Product”は、プログラムの集まりを意味し、このプログラムの集まりは、通常、業務単位に設定されるものであり、“Product”に引き続きプロダクト名を記述する。個々の“Product”タグ内には、複数の“Program”タグを記述することができる。これは、“Product”に含まれるアプリケーション・プログラム毎に、固有のバージョン・チェック方法を提供するためのものである。

【0036】クライアント・マシン2側のアクション記述ファイル11の構成は、該当するプロダクトに関してバージョンチェック前の条件及び不一致検出後の処理に関するデフォルトの動作を規定する部分213と、あるプロダクト中に含まれるプログラムのうちデフォルトの対処方法を用いないものに関する記述を行なう部分214とからなる。

【0037】サーバ・マシン1側のアクション記述ファイル11の構成は、該当するプロダクトに関して、バ-

10

ジョン不一致のチェック及びその後の動作に関するデフォルトの動作を規定する部分311と、あるプロダクト中に含まれるプログラムのうちデフォルトのチェック方法を用いないものに関する記述を行なう部分312とからなる。本発明によるバージョン管理方法は、プロダクト/プログラム何れの単位での管理とも可能である。

【0038】次に、図4は本発明に係る実施の形態1のネットワークシステムのソフトウェア・バージョン管理方式におけるシステム動作フローを示すフローチャート10である。サーバ・マシン1側のバージョン管理サーバ・プログラム8は、任意のタイミングでクライアント・マシン2からのバージョンチェック要求を受け付けるために常に動作し、クライアント・マシン2からのバージョンチェック要求を待つ。クライアント・プログラム13の起動は、バージョン管理クライアント・プログラム12を通して行なわれる。

【0039】クライアント・マシン2のバージョン管理クライアント・プログラム12は、起動されると、まず、アクション記述ファイル11を読み込む（ステップS1）。バージョン管理クライアント・プログラム12は、アクション記述ファイル11を読み込んだ結果、アプリケーション実行のためのバージョンチェックが必要であると判断した場合（ステップS2）、自プログラムのバージョン番号、アプリケーション名、プロダクト名等のバージョンチェックに必要なバージョン情報を設定し、設定したバージョン情報をサーバ・マシン1のバージョン管理サーバ・プログラム8へを転送して、バージョンチェック要求を行なう（ステップS3）。

【0040】サーバ・マシン1のバージョン管理サーバ・プログラム8は、クライアント・マシン2から転送されるバージョン情報を受信して、バージョン・チェック要求を受け付ける（ステップS4）、サーバ・マシン1上のアクション記述ファイル9を読み込む（ステップS5）。そして、バージョン管理サーバ・プログラム8は、アクション記述ファイル9内に記述されたチェック方法に従い、クライアント・プログラム13とサーバ・プログラム14間のバージョン・チェック処理を行ない（ステップS6）、クライアント・プログラム13とサーバ・プログラム14が同一バージョンであるか否かのバージョンチェック結果を、バージョンチェック要求元のクライアント・マシン2のバージョン管理クライアント・プログラム12へ返す（ステップS7）。

【0041】クライアント・マシン2のバージョン管理クライアント・プログラム12は、サーバ・マシン1から転送されるバージョンチェック結果を受信し（ステップS8）、クライアント・プログラムとサーバ・プログラムのバージョンが一致である旨のバージョンチェック結果の通知を受けた場合（ステップS9）、引き続きアプリケーションクライアント・プログラム13を起動する（ステップS10）。

11

【0042】一方、クライアント・マシン2のバージョン管理クライアント・プログラム12は、クライアント・プログラム13とサーバ・プログラム14のバージョンが一致していないと判断した場合（ステップS9）、直ちにソフトウェアの入れ替えを実施するか否かを判断する（ステップS11）。クライアント・マシン2のバージョン管理クライアント・プログラム12は、直ちにソフトウェアの入れ替えを実施すると判断した場合（ステップS11）、アクション記述ファイル11に従ったソフトウェアの入れ替えを実施し（ステップS12）、アプリケーションクライアント・プログラム13を起動する（ステップS13）。

【0043】クライアント・マシン2のバージョン管理クライアント・プログラム12は、直ちにソフトウェアの入れ替えを実施しないと判断した場合（ステップS11）、クライアント・プログラム13の起動を実施するか否かを判断する（ステップS14）。クライアント・マシン2のバージョン管理クライアント・プログラム12は、クライアント・プログラム13の起動を実施すると判断した場合（ステップS14）、クライアント・プログラム13の起動を実施し（ステップS15）、アクション記述ファイル11に従ったソフトウェア入れ替えを指示する（ステップS16）。

【0044】このように、本実施の形態では、クライアント・マシン2側で、バージョンチェックに必要なクライアント・プログラム13のバージョン情報を設定し、設定したバージョン情報をサーバ・マシン1へ転送してバージョンチェック要求を行ない、サーバ・マシン1側で、クライアント・マシン2から転送されるクライアント・プログラム13のバージョン情報を受信してバージョンチェック要求を受け付け、受信したクライアント・プログラム13のバージョン情報とバージョン管理データベース6から読み出したサーバ・プログラム14のバージョンチェックを行なった後、そのバージョンチェック結果をクライアント・マシン2へ転送するように構成している。更に、本実施の形態では、クライアント・マシン2側で、サーバ・マシン1から転送されるバージョンチェック結果に基づいて、クライアント・プログラム13とサーバ・プログラム14のバージョンが一致していないかを判断し、一致していないと判断した場合、バージョンが一致していないクライアント・プログラム13をサーバ・プログラム14のバージョンと一致するように入れ替え、入れ替えたクライアント・プログラム13を実行するように構成している。このため、クライアント・プログラム13を実行する時、実行するクライアント・プログラム13のバージョンをサーバ・プログラム14のバージョンと確実に合わせることができる。クライアント・プログラム13実行時にバージョン不一致による不整合を起こらないようにすることができ、安定したシステム動作を行なうことができる。従つ

12

て、常にバージョンの一致した環境での処理を保証することができる。

【0045】また、本実施の形態では、クライアント・マシン2側でバージョン不一致だったクライアント・プログラム13を入れ替えるように構成したため、従来の配布サーバからソフトウェアを貰って入れ替える場合よりも、ネットワークエラーが生じても確実にクライアント・プログラム13を入れ替えることができる。

【0046】実施の形態2. 本実施の形態は、請求項2/10の発明に係る特徴部分のみ説明し、図1、4を参照して説明する。上記実施の形態1では、クライアント・マシン2側でクライアント・プログラム13のバージョン情報を設定した後、直ちに設定したバージョン情報をサーバ・マシン1へ転送してバージョンチェック要求を行なう場合を説明したが、本実施の形態では、クライアント・マシン2のアクション記述ファイル11にバージョンチェックのタイミングを指定しておき、バージョン管理クライアント・プログラム12により、指定したバージョンチェックのタイミングに達したと判断した場合、設定したクライアント・プログラム13のバージョン情報をサーバ・マシン1へ転送してバージョンチェック要求を行なうように構成する。この場合、指定したタイミングでバージョンチェックを行なうことができるので、バージョンチェックを効率よく行なうことができる。以下、具体的に図面を用いて説明する。

【0047】図5は本発明に係る実施の形態2の図1に示すアクション記述ファイル11におけるバージョンチェックタイミング設定に関する記述例を示す図である。アプリケーション実行環境において、クライアント／サーバ間でソフトウェアのバージョンが一致しているか否かを確実にチェックするためには、アプリケーションの起動のタイミングでバージョンのチェックを行なえばよい。

【0048】しかしながら、アプリケーションによっては、常に、バージョンのチェックを行なう必要がないことも多い。本実施の形態では、このアプリケーション起動時のバージョン・チェック処理を実施するタイミングについて、アクション記述ファイル11に従って行なうことを探としている。アクション記述ファイル11には、プログラム起動毎（every）／1日の最初の一回（1st）／チェック未実施（none）の何れかの指定を行なう。

【0049】このアクション記述ファイル11のアプリケーション2（app. 2）に関しては、プログラム起動毎にチェックを実施する設定502であり、アプリケーション1（app. 1）に関しては、一日の内で最初の一回目だけチェックを実施する設定501である。この設定の行なわれた状態でのバージョン管理クライアント・プログラム13の動作を、図4のフローチャートを基に説明する。

13

【0050】クライアント・マシン2のバージョン管理
クライアント・プログラム12は、その起動時に、アクション記述ファイル11を読み込み、バージョンチェックを実施するかの情報を、Checkフィールドより得る。その結果、バージョン管理クライアント・プログラム12は、例えば、図5のアクション記述ファイル11の設定で、app.2を起動した時のように、バージョンチェックの実施が必要であると判断した場合（ステップS2）、設定したクライアント・プログラムのバージョン情報をサーバ・マシン1のバージョン管理サーバ・プログラム8へ転送して、バージョンチェック要求を行なう（ステップS3）。

【0051】一方、クライアント・マシン2のバージョン管理クライアント・プログラム12は、例えば図5の設定で、app.1を連続して何回か起動した時の2回目以降の起動時のようにバージョンチェックを行なう必要がないと判断した場合（ステップS2）、サーバ・マシン1のバージョン管理サーバ・プログラム8へのバージョンチェック要求は発行せず、クライアント・プログラム13の起動を行なう（ステップS10）。

【0052】このように、本実施の形態では、クライアント・マシン1のアクション記述ファイル11にバージョンチェックのタイミングを指定しておき、バージョン管理クライアント・プログラム12により、指定したバージョンチェックのタイミングに達したと判断した場合、設定したクライアント・プログラムのバージョン情報をサーバ・マシン1へ転送してバージョンチェック要求を行なうように構成することにより、指定したタイミングでバージョンチェックを行なうことができるので、バージョンチェックを効率よく行なうことができる。例えば、アプリケーション起動の度にバージョンチェックの実施やソフトウェアのダウンロードを行なわないで済ませることができるので、このため、システムの運用上不必要的バージョンチェックを行なわないので済ませるので、性能の劣化や不用意な負荷の増大を防ぐことができる。

【0053】実施の形態3. 本実施の形態は、請求項3の発明に係る特徴部分のみを説明し、図1、4、6を参照して説明する。上記実施の形態1では、バージョン管理サーバ・プログラム8により、クライアント・マシン2からバージョンチェック要求を受け付けると、受信したクライアント・プログラムのバージョン情報とバージョン管理データベース6から読み出したサーバ・プログラムのバージョン情報を比較してバージョンチェックを行なう場合を説明したが、本実施の形態では、バージョンチェックを行なう前に、サーバ・マシン1のアクション記述ファイル9にバージョンチェックに用いるバージョンチェック情報を設定しておき、バージョン管理サーバ・プログラム8により、アクション記述ファイル9から読み出したバージョンチェック情報を基づいてバ-

14

ジョンチェックを行なう（ステップS5、S6）ように構成する。この場合、設定したバージョンチェック情報に基づいてバージョンチェックを行なうことができるので、ソフトウェアに合わせた柔軟なバージョン解釈方法を指定することができる。以下、具体的に図面を用いて説明する。

【0054】図6は本発明に係る実施の形態3の図1に示すアクション記述ファイル9の内容を示す図である。

ソフトウェアのバージョンをチェックする際、チェックの基準となる情報としては大きく、このチェック基準となる情報には、ファイルの作成日付（“date”）、ファイル名（例えば、バージョンを含んだプロダクト名の場合）、ファイルやモジュール中に埋め込まれたバージョン情報（“revision”）、そして、バージョン管理データベース6への登録内容（“config”）、例えば、プロダクトやアプリケーション群をまとめて版管理する場合）等が挙げられる。

【0055】しかしながら、バージョンチェックの基準となる情報は多いものの、それらを結合してバージョンチェックを行なうのは困難である。本実施の形態は、

“Information”フィールド611、616にファイル作成日付（“date”）／ソフトウェアに埋め込まれた版情報（“revision”）／バージョン管理データベース6の登録内容（“config”）の何れかを記述することにより、バージョンチェックに用いる情報を設定する。

【0056】例えば、図6では、プロダクトPRO1に関して、Informationフィールドに“config”が記述されていることから、バージョン管理データベース6を基にしたバージョン・チェックを優先することになる。バージョン管理サーバ・プログラム8は、このInformationに記述の情報を基に、バージョンの比較を行なう。図6の例の場合、例えばクライアント・プログラム13とサーバ・プログラム14のモジュールの中に含まれたバージョン（revision）に相違があっても、バージョン管理データベース6の登録内容に相違がなければ、クライアント・プログラム13とサーバ・プログラム14間にバージョンの相違はないものと判断される。

【0057】このように、本実施の形態では、バージョンチェックを行なう前に、サーバ・マシン1のアクション記述ファイル9にバージョンチェックに用いるバージョンチェック情報を設定しておき、バージョン管理サーバ・プログラム8により、この設定したバージョンチェック情報に基づいてバージョンチェックを行なうように構成することにより、設定したバージョンチェック情報に基づいてバージョンチェックを行なうことができるので、ソフトウェアに合わせた柔軟なバージョン解釈方法を指定することができる。このため、ソフトウェアによって異なるバージョン情報に対応することができるの

15

で、効率の良いシステム運用を実現することができる。
【0058】実施の形態4. 本実施の形態は、請求項4の発明に係る特徴部分のみを説明し、図1、4、6を参照して説明する。上記実施の形態1では、バージョン管理サーバ・プログラム8により、クライアント・マシン2からバージョンチェック要求を受け付けると、受信したクライアント・プログラムのバージョン情報とバージョン管理データベース6から読み出したサーバ・プログラムのバージョン情報を比較してバージョンチェックを行なう場合を説明したが、本実施の形態では、バージョンチェックを行なう前に、アクション記述ファイル9にバージョンチェック時のチェックロジック情報を設定しておき、バージョン管理サーバ・プログラム8により、アクション記述ファイル9から読み出したチェックロジック情報に基づいてバージョンチェックを行なう

(ステップS5、S6) ように構成する。この場合、設定したチェックロジック情報に基づいてバージョンチェックを行なうことができるので、ソフトウェアに合わせた柔軟なバージョン解釈方法を指定することができる。

【0059】本実施の形態は、バージョン・チェックのアルゴリズムに関するものである。実際のシステムでは、ネットワーク環境内に分散したクライアント/サーバ型アプリケーションのバージョンは、同一プロダクトに関しては一致していることが望ましい。しかしながら、実際のシステム環境では、サーバ・マシンにおける24時間連続運転実施の場合等、容易にソフトウェアの入れ替えを実施できない環境もある。こういった環境内でクライアント/サーバ間のアプリケーションのバージョンが異なってしまった場合、ソフトウェアの入れ替えを極力少なくするために、バージョンの完全な一致はないものの、動作上問題が発生しないことが判つていれば、バージョン一致とみなすことが望ましい。

【0060】本実施の形態では、サーバ・マシン1のアクション記述ファイル9に“Algorithm”フィールド611、616を設け、クライアント・プログラム13とサーバ・プログラム14の間で、完全にバージョンの一致が必要(“complete”)／特定の範囲内のバージョンであればバージョンが一致したとみなす(“range”)の何れかを記述することにより、バージョンチェックの考え方(完全一致又は部分一致)に関する設定を行なう。

【0061】このように、本実施の形態では、バージョン管理サーバ・プログラム8により、アクション記述ファイル9に設定したチェックロジック情報に基づいてバージョンチェックを行なうように構成することにより、ソフトウェアに合わせた柔軟なバージョン解釈方法を指定することができる。このため、バージョンの異なるソフトウェアが混在する環境等、単純なバージョン比較のみでは対応できない複雑な環境化においても、バージョンチェック・アルゴリズムをカスタマイズすることによ

り、対応することができる。

【0062】実施の形態5. 本実施の形態は、請求項5の発明に係る特徴部分のみを説明し、図1、4、6を参照して説明する。上記実施の形態1では、バージョン管理サーバ・プログラム8により、クライアント・マシン2からバージョンチェック要求を受け付けると、受信したクライアント・プログラム13のバージョン情報とバージョン管理データベース6から読み出したサーバ・プログラム14のバージョン情報を比較してバージョンチェックを行なう場合を説明したが、本実施の形態では、バージョンチェックを行なう前に、アクション記述ファイル9にバージョンチェック時のチェックロジック情報を設定しておき、バージョン管理サーバ・プログラム8により、アクション記述ファイル9から読み出した関連するソフトウェア間の依存関係情報をに基づいてバージョンチェックを行なう(ステップS5、S6) ように構成する。この場合、設定した関連するソフトウェア間の依存関係情報に基づいてバージョンチェックを行なうことができるので、依存関係のあるソフトウェア全体のバージョンチェックを一括して行なうことができる。

【0063】本実施の形態は、ソフトウェアの依存関係を考慮したバージョン・チェック方法に関するものである。分散システム環境内では、相互に依存関係にあるアプリケーションが多く存在する。例えば、プロダクト1に含まれるアプリケーションとして、クライアント・プログラムclient1及びサーバ・プログラムserver2が存在し、プロダクト2に含まれるアプリケーションとして、クライアント・プログラムclient2及びサーバ・プログラムserver2が存在した場合を考える。

【0064】ここで、プロダクト2に含まれるアプリケーション・プログラムだけが分散システム環境内に配信された時、プロダクト1に含まれるクライアント・プログラムclient1及びサーバ・プログラムserver1間のバージョンは、一致しているにも拘らず、クライアント・プログラムclient1を実行することができないか、若しくは、実行結果に誤りを生じるといった問題が発生することがある。これは、プロダクト1内のソフトウェアがプロダクト2内の機能を使っているために発生するものである。

【0065】本実施の形態では、図6に示すように、アクション記述ファイル9にフィールド“dependency”613、618を設け、プロダクトPRO1とプロダクトPRO2間の依存関係情報を記述しておくことにより、プロダクトPRO1中のSERVER1に対する要求発行時、起動時のバージョンのチェックとして、プロダクトPRO1だけでなく、プロダクトPRO2中のバージョンとの比較処理を行なうことにより、バージョン一致をチェックする。

【0066】このように、本実施の形態では、バージョ

17

ン管理サーバ・プログラム8により、アクション記述ファイル9に設定した関連するソフトウェア間の依存関係情報に基づいてバージョンチェックを行なうように構成することにより、管理単位を跨って依存関係を有するソフトウェアに対して、相互の依存関係を指定することができるので、効率の良いシステム運用を実現することができる。本実施の形態では、1つのソフトウェアを入れ替える際、そのソフトウェアと依存関係にあるソフトウェアを入れ替えることができるので、システム管理者の負担を軽減することができる。

【0067】実施の形態6. 本実施の形態は、請求項6の発明に係る特徴部分のみを説明し、図1, 4, 7を参照して説明する。上記実施の形態1では、バージョン管理クライアント・プログラム12により、クライアント・プログラム13とサーバ・プログラム14のバージョンが一致していない時（ステップS9）、直ちにソフトウェアの入れ替えを実施すると判断した場合（ステップS11）、ソフトウェアの入れ替えを実施する場合を説明したが、本実施の形態では、クライアント・プログラム13を入れ替える前に、アクション記述ファイル11にクライアント・プログラム13を入れ替えるタイミングを設定しておき、バージョン管理クライアント・プログラム12により、設定したクライアント・プログラム13の入れ替えタイミングに達した時（ステップS11）、クライアント・プログラム13を入れ替えるように構成する。この場合、設定したクライアント・プログラム13の入れ替えタイミングに達した時に、クライアント・プログラム13を入れ替えることができるので、システム運用に合わせたソフトウェア入れ替えを実施することができる。

【0068】本実施の形態は、バージョンチェック実施の結果、クライアント・プログラム13のバージョン情報とサーバ・プログラム14のバージョン情報に相違が発生した場合のソフトウェア入れ替え方法に関する。バージョンチェックの結果、分散システム環境においてアプリケーションを動作させるための正しいバージョンでないことが判明した場合、通常はアプリケーションを入れ替えることが望ましい。ここで、このアプリケーションの入れ替え作業は、クライアント・マシン2の台数が多くったり、システムが動作中である場合には、特に困難であり、システム環境に応じたソフトウェア入れ替え方法により、バージョン不整合を解消することが望ましい。

【0069】本実施の形態は、アクション記述ファイル11の記述内容に従ったバージョンチェックに失敗時のバージョン不一致となったソフトウェアの入れ替え方法に関する。このアクション記述ファイル11の記述は、図2の212部分に相当する。図7は図2に示す212部分の具体的な記述例を示す図である。本実施の形態は、バージョンチェックの結果、ソフトウェアの入れ替

18

えが必要と判断された際、実際のプログラムの入れ替え作業を実施するタイミングを指定する手段を設けるものである。入れ替えタイミングとしては、バージョンチェックによる不一致検出直後（“now”）、不一致検出後もアプリケーションは動作しアプリケーション自身の終了直後（“after”）、時刻指定による入れ替え（“time”、業務終了時刻の入れ替え等）を設定することができる。“now”、“after”、“time”何れかの情報は、サーバ・マシン1のアクション記述ファイル11の“when”フィールドに記述する。

【0070】図7のアクション記述ファイル11の記述例におけるプログラムapp-aに関しては、バージョン不一致を検出した時点でアプリケーションの実際の処理を行なわず即座に入れ替え711を行なうための設定であり、プログラムapp-bに関しては、深夜時間（業務時間外）の入れ替え（717）を行なうための設定である。図4, 7を基に、動作を説明する。

【0071】クライアント・マシン2のバージョン管理20 クライアント・プログラム12は、起動されてバージョンチェックの不一致を検出した際、図4のステップ9の部分の処理で入れ替え条件を判断する。そして、バージョン管理クライアント・プログラム12は、アクション記述ファイル11の記述として、図7のプロダクトPRO-Xに属するようなすぐに入れ替えが必要な場合（ステップS11）、配信プログラムに対して、必要なソフトウェアの条件を通知し、クライアント・プログラム13のソフトウェアの入れ替えを行なう（ステップS12）。

【0072】クライアント・マシン2のバージョン管理 クライアント・プログラム12は、入れ替え環境を確認した上で、入れ替えたクライアント・プログラム13のアプリケーションを起動する（ステップS13）。一方、バージョン管理クライアント・プログラム12は、アクション記述ファイル11の記述として、図7のプロダクトPRO-Yに属するような業務時間外入れ替えが指定された場合、バージョン管理クライアント・プログラム12は、システムの提供するスケジュール機能等へ必要なアプリケーション配布処理の登録を行なう（ステップ16）。登録内容としては、配布時間、配布対象プロダクトの指定が挙げられる。

【0073】このように、本実施の形態では、クライアント・プログラム13を入れ替える前に、アクション記述ファイル11にクライアント・プログラム13を入れ替えるタイミングを設定しておき、バージョン管理クライアント・プログラム12により、設定したクライアント・プログラム13の入れ替えタイミングに達した時（ステップS11）、クライアント・プログラム13を入れ替えるように構成したため、設定したクライアント・プログラム13の入れ替えタイミングに達した時に、

19

クライアント・プログラム13を入れ替えることができる、システム運用に合わせたソフトウェア入れ替えを実施することができる。

【0074】本実施の形態では、例えば、今、ソフトウェアを直ぐに入れ替えたいが入れ替えられない時、夜間に入れ替えたり、一連の業務が終了した後に入れ替えたりすることを、予め指定することができる。このため、システム要件に合わせて柔軟にソフトウェアの入れ替えスケジュールを設定することができるので、効率の良いシステム運用を実現することができる。

【0075】実施の形態7. 本実施の形態は、請求項7の発明に係る特徴部分のみを説明し、図1、4、7を参照して説明する。上記実施の形態1では、バージョン管理クライアント・プログラム12により、クライアント・プログラム13とサーバ・プログラム14のバージョンが一致していない時（ステップS9）、直ちにソフトウェアの入れ替えを実施すると判断した場合（ステップS11）、ソフトウェアの入れ替えを実施する場合を説明したが、本実施の形態では、クライアント・プログラム13を入れ替える前に、クライアント2のアクション記述ファイル11にクライアント・プログラム13の入れ替え条件を設定しておき、バージョン管理クライアント・プログラム12により、設定したクライアント・プログラム13の入れ替え条件を基にクライアント・プログラム13を入れ替えるように構成する。この場合、設定したクライアント・プログラム13の入れ替え条件を基にクライアント・プログラム13を入れ替えることができるので、例えば、重要度の高いものはバージョンチェックを頻繁に行なうといった、環境や業務内容に応じた柔軟なソフトウェア入れ替えを実現することができる。

【0076】本実施の形態は、バージョンチェックの結果、クライアント・プログラム13の入れ替えが必要であると判断された際、事前に設定されたクライアント・プログラム13の入れ替え条件に応じて入れ替えを行なうものである。ソフトウェアの入れ替えを行なう際は通常、入れ替え対象となるソフトウェアが動作状態では実施できない。そこで、実システムにおいて業務実施中の場合、この入れ替え作業を実施するのは、時間的に困難な時もある。業務実施中か処理に関わらず、緊急に入れ替えが必要なことが多い。それは例えば、頻繁に使用されるソフトウェア、重要作業を行なうような特定の端末、非常に重要な特定の業務といった条件があつてまる。

【0077】本実施の形態では、クライアント・マシン2のアクション記述ファイル11に“Whether”フィールド712、717を設け、サーバ・マシン1のバージョン管理サーバ・プログラム8からのバージョン・チェック応答により、バージョンが不一致という通知を受けた際も、アクション記述ファイル11の“Whether”

20

”フィールドの内容に応じて、クライアント・マシン2のバージョン管理クライアント・プログラム12は、不一致だったクライアント・プログラム12の入れ替え処理を実施する（ステップS11、S12）。例えば図7では、プロダクトPRO-Xに関しては、バージョン不一致回数（times）の情報を記述することにより、同一アプリケーションについてN回以上のバージョン相違が検出された時点で、クライアント・プログラム13のソフトウェアの入れ替えを実施する。

10 【0078】このように、本実施の形態では、クライアント・プログラム13を入れ替える前に、クライアント2のアクション記述ファイル11にクライアント・プログラム13の入れ替え条件を設定しておき、バージョン管理クライアント・プログラム12により、設定したクライアント・プログラム13の入れ替え条件を基にクライアント・プログラム13を入れ替えるように構成したため、設定したクライアント・プログラム13の入れ替え条件を基にクライアント・プログラム13を入れ替えることができる。このため、例えば、重要度の高いものはバージョンチェックを頻繁に行なうといった、環境や業務内容に応じた柔軟なソフトウェア入れ替えを実現することができる。

【0079】実施の形態8. 本実施の形態は、請求項8の発明に係る特徴部分のみを説明し、図1、4、7を参照して説明する。上記実施の形態1では、バージョン管理クライアント・プログラム12により、クライアント・プログラム13とサーバ・プログラム14のバージョンが一致していない時（ステップS9）、直ちにソフトウェアの入れ替えを実施すると判断した場合（ステップS11）、ソフトウェアの入れ替えを実施する（ステップS12）場合を説明したが、本実施の形態は、バージョン・チェックの結果、クライアント・プログラム13の入れ替えが必要と判断された際、入れ替え対象のクライアント・プログラム13を選択する手段を設けるものである。分散システム環境内の各クライアント・マシン2でクライアント・プログラム13のバージョンの相違が検出され、クライアント・プログラム13の入れ替えが必要だと判断された際、クライアント・プログラム13の入れ替え作業に伴う稼動中の業務への影響を最小限に抑えるためには、ある特定の業務を構成する全てのソフトウェア群を一括して入れ替えるのではなく、必要最小限のアプリケーション・プログラムのみ入れ替えることが望ましい。

40 【0080】本実施の形態では、アクション記述ファイル11に“What”フィールド713、718を設け、マシン間でプロダクトの不一致が検出された際、該当するプロダクトの中から“What”フィールドに記述されたモジュール（実行形式のファイル）のみの入れ替えを行なう。例えば、図7において、プロダクトPRO-Xが100個のモジュールより構成されていたと

21

し、バージョン不一致によるクライアント・プログラム13の入れ替えが必要となった際、100個のモジュール全てを入れ替えるのではなく、“What”フィールドに記述の3つのモジュール“prog_1”、“prog_2”、“prog_3”のみの入れ替えを実施する。

【0081】このように、本実施の形態では、クライアント・プログラム13を入れ替える前に、クライアント・マシン2のアクション記述ファイル11に入れ替え対象のクライアント・プログラム13を選択する条件を設定しておき、バージョン管理クライアント・プログラム12により、設定した入れ替え対象のクライアント・プログラム13を基にクライアント・プログラム13を入れ替えるように構成する。この場合、設定した入れ替え対象のクライアント・プログラム13を基にクライアント・プログラム13を入れ替えることができるので、関連するクライアント・プログラム13を一括して入れ替えるのではなく、モジュール単位で入れ替えを行なうことができる。従って、システムや業務要件に合わせたモジュール単位での入れ替えを行なうことができるので、不要なモジュールの入れ替えを極力行なわない効率のよいシステム運用を実現することができる。

【0082】

【発明の効果】請求項1記載の発明によれば、クライアント側で、バージョンチェックに必要なクライアント・プログラムのバージョン情報を設定し、設定したバージョン情報をサーバへ転送してバージョンチェック要求を行ない、サーバ側で、クライアントから転送されるクライアント・プログラムのバージョン情報を受信してバージョンチェック要求を受け付け、受信したクライアント・プログラムのバージョン情報とバージョン管理データベースから読み出したサーバ・プログラムのバージョンチェックを行なった後、そのバージョンチェック結果をクライアント・マシンへ転送するように構成し、更に、クライアント側で、サーバから転送されるバージョンチェック結果に基づいて、クライアント・プログラムとサーバ・プログラムのバージョンが一致していないかを判断し、一致していないと判断した場合、バージョンが一致していないクライアント・プログラムをサーバ・プログラムのバージョンと一致するように入れ替え、入れ替えたクライアント・プログラムを実行するように構成したため、クライアント・プログラムを実行する時、実行するクライアント・プログラムのバージョンをサーバ・プログラムのバージョンと確実に合わせることができる。従って、常にバージョンの一一致した環境での処理を保証することができる。

【0083】また、請求項1記載の発明によれば、クライアント側でバージョン不一致だったクライアント・プログラムを入れ替えるように構成したため、従来の配布サーバからソフトウェアを貰って入れ替える場合よりも、ネットワークエラーが生じても確実にクライアント・プログラムを入れ替えることができる。

22

【0084】請求項2記載の発明によれば、クライアントにバージョンチェックのタイミングを指定しておき、指定したバージョンチェックのタイミングに達したと判断した場合、設定したクライアント・プログラムのバージョン情報をサーバへ転送してバージョンチェック要求を行なうように構成することにより、指定したタイミングでバージョンチェックを行なうことができるので、バージョンチェックを効率よく行なうことができる。例えば、アプリケーション起動の度にバージョンチェックの実施やソフトウェアのダウンロードを行なわないで済ませることができる。このため、システムの運用上不必要なバージョンチェックを行なわないで済ませることができるので、性能の劣化や不用意な負荷の増大を防ぐことができる。

【0085】請求項3記載の発明によれば、バージョンチェックを行なう前に、サーバにバージョンチェックに用いるバージョンチェック情報を設定しておき、この設定したバージョンチェック情報に基づいてバージョンチェックを行なうように構成することにより、設定したバージョンチェック情報に基づいてバージョンチェックを行なうことができるので、ソフトウェアに合わせた柔軟なバージョン解釈方法を指定することができます。このため、ソフトウェアによって異なるバージョン情報に対応することができるので、効率の良いシステム運用を実現することができる。

【0086】請求項4記載の発明によれば、サーバに設定したチェックロジック情報に基づいてバージョンチェックを行なうように構成することにより、ソフトウェアに合わせた柔軟なバージョン解釈方法を指定することができる。このため、バージョンの異なるソフトウェアが混在する環境等、単純なバージョン比較のみでは対応できない複雑な環境化においても、バージョンチェック・アルゴリズムをカスタマイズすることにより、対応することができる。

【0087】請求項5記載の発明によれば、サーバに設定した関連するソフトウェア間の依存関係情報に基づいてバージョンチェックを行なうように構成することにより、管理単位を跨って依存関係を有するソフトウェアに対して、相互の依存関係を指定することができるので、効率の良いシステム運用を実現することができる。例えば、1つのソフトウェアを入れ替える際、そのソフトウェアと依存関係にあるソフトウェアを入れ替えることができるので、システム管理者の負担を軽減することができる。

【0088】請求項6記載の発明によれば、クライアント

23

ト・プログラムを入れ替える前に、クライアントにクライアント・プログラムを入れ替えるタイミングを設定しておき、設定したクライアント・プログラムの入れ替えタイミングに達した時、クライアント・プログラムを入れ替えるように構成したため、設定したクライアント・プログラムの入れ替えタイミングに達した時に、クライアント・プログラムを入れ替えることができるので、システム運用に合わせたソフトウェア入れ替えを実施することができる。

【0089】請求項7記載の発明によれば、クライアント・プログラムを入れ替える前に、クライアントにクライアント・プログラムの入れ替え条件を設定しておき、設定したクライアント・プログラムの入れ替え条件を基にクライアント・プログラムを入れ替えるように構成したため、設定したクライアント・プログラムの入れ替え条件を基にクライアント・プログラムを入れ替えることができる。このため、例えば、重要度の高いものはバージョンチェックを頻繁に行なうといった、環境や業務内容に応じた柔軟なソフトウェア入れ替えを実現することができる。

【0090】請求項8記載の発明によれば、クライアント・プログラムを入れ替える前に、クライアントに入れ替え対象のクライアント・プログラムを選択する条件を設定しておき、設定した入れ替え対象のクライアント・プログラムを基にクライアント・プログラムを入れ替えるように構成したため、設定した入れ替え対象のクライアント・プログラムを基にクライアント・プログラムを入れ替えることができるので、関連するクライアント・プログラムを一括して入れ替えるのではなく、モジュール単位で入れ替えを行なうことができる。従って、システムや業務要件に合わせたモジュール単位での入れ替えを行なうことができるので、不要なモジュールの入れ替えを極力行なわない効率のよいシステム運用を実現することができる。

24

【図面の簡単な説明】

【図1】 本発明に係る実施の形態1のネットワークシステムのソフトウェア・バージョン管理方式のシステム構成を示すブロック図である。

【図2】 図1に示すクライアント・マシン上のアクション記述ファイルの構成を示す図である。

【図3】 図1に示すサーバ・マシン上のアクション記述ファイルの構成を示す図である。

【図4】 本発明に係る実施の形態1のネットワークシステムのソフトウェア・バージョン管理方式におけるシステム動作フローを示すフローチャートである。

【図5】 本発明に係る実施の形態2のアクション記述ファイルにおけるバージョンチェックタイミング設定に関する記述例を示す図である。

【図6】 本発明に係る実施の形態3のアクション記述ファイルの内容を示す図である。

【図7】 本発明に係る実施の形態6のアクション記述ファイルの内容を示す図である。

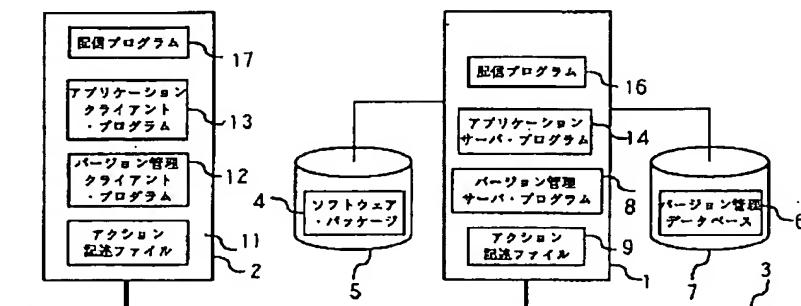
【図8】 従来のソフトウェアのバージョン管理方式におけるシステム動作フローを示すフローチャートである。

【図9】 従来のソフトウェアのバージョン管理方式におけるシステム動作フローを示すフローチャートである。

【符号の説明】

- 1 サーバ・マシン、2 クライアント・マシン、3 通信媒体、4 ソフトウェア・パッケージ、5, 7 磁気ディスク装置、6 バージョン管理データベース、8 バージョン管理サーバ・プログラム、9, 11 アクション記述ファイル、12 バージョン管理クライアント・プログラム、13 クライアント・マシン上のアクション記述ファイル、14 サーバ・マシン上のアクション記述ファイル、16 サーバ・マシン上のバージョン管理データベース、17 クライアント・マシン上のバージョン管理データベース。

【図1】



【図2】

product: プロダクト名(1) 11
 221 パーバージョンチェック前の条件を記述り
 Check: パーバージョン不一致か否かをチェックする時に使うする既定り 211

 222 パーバージョンチェック不一致検出後の処理を記述り 212
 When: リポジトリストアル場所に因する既定り 213
 Whether: フォルダの入れ替えを行なう条件に因する既定り 213
 What: フォルダの入れ替えを行なうソフトウェアに因する既定り 213

 program: プログラム(1), プログラム(2), ..., プログラム(N)
 222 Check: When: Whether: What:
 に因して、必要な定義を記述する。 214

 program: プログラム(A), プログラム(B), ..., プログラム(C)
 222 Check: When: Whether: What:
 に因して、必要な定義を記述する。 214

 product: プロダクト名(2) 213
 221

【図3】

product: プロダクト名(1) 9
 321 Information: パーバージョン比較の判断に用いる情報に関する既定り 311
 Algorithm: パーバージョン一数の考え方に関する既定り 311
 Dependency: 依存するプログラムやプロダクトの情報の記述り 311

 program: プログラム(1), プログラム(2), ..., プログラム(N)
 322 Information: Algorithm: Dependency: 312
 322 に因して、必要な定義を記述する。

 program: プログラム(A), プログラム(B), ..., プログラム(C)
 322 Information: Algorithm: Dependency: 312
 322 に因して、必要な定義を記述する。

 product: プロダクト名(2)
 321

【図5】

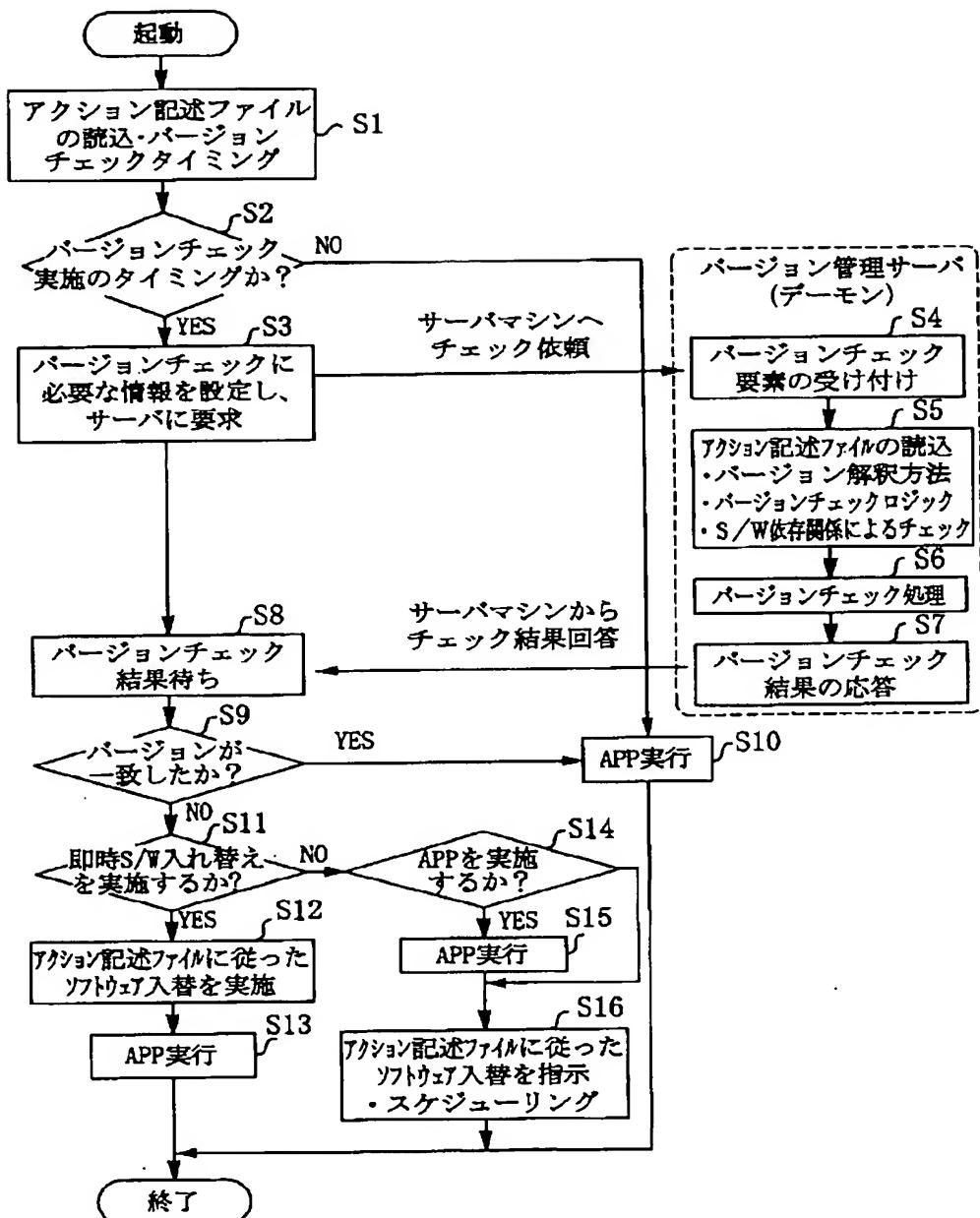
program: app.1
 501 Check: 1st /* 1st, every, none, 等指定可 */
 program: app.2
 502 Check: every /* 1st, every, none, 等指定可 */

【図6】

601 Product: PRO1 /* プロダクト名 */
 601 Information: config /* config, revision, date, 等指定可 */
 611 Algorithm: complete /* complete, range, 等指定可 */
 612 Dependency: PRO2 /* 他プロダクト or プログラム名指定 */
 613 Program: server1,app.1,... /* プロダクトに含まれるプログラムを列記する。 */
 602

 606 Product: PRO2 /* プロダクト名 */
 606 Information: config /* config, revision, date, 等指定可 */
 616 Algorithm: range /* complete, range, 等指定可 */
 617 Dependency: /* 他プロダクト or プログラム名指定 */
 618 Program: server2,app.2,... /* プロダクトに含まれるプログラムを列記する。 */
 607

【図4】



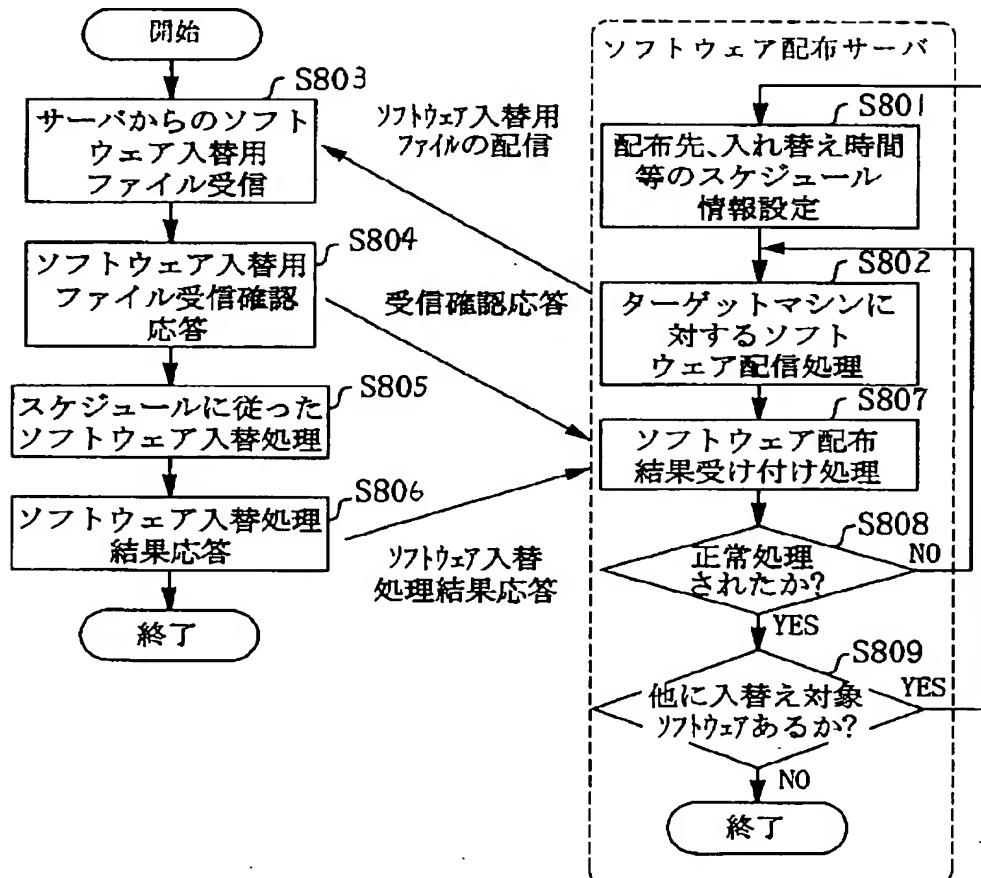
【図7】

```

product: PRO-X
701 When: now
711 Whether: times: N
712 What: prog_1, prog_2, prog_3
713 program: app-a
702

product: PRO-Y
706 When: time: 00:00
716 Whether:
717 What: prog_7
718 program: app-b
707
  
```

【図8】



【図9】

